IBM DB2 Utilities Enhancement Tool for z/OS User's Guide

Version 2 Release 2

SC19-3417-05

IBM DB2 Utilities Enhancement Tool for z/OS User's Guide

Version 2 Release 2

SC19-3417-05

Note:
Before using this information and the product it supports, read the "Notices" topic at the end of this information.

Sixth Edition (April 2016)

This edition applies to Version 2 Release 2 of IBM DB2 Utilities Enhancement Tool for z/OS (product number 5655-T58) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC19-3417-04.

Contents

Figures

Tables

About this information

Throughout this information, the DB2® Utilities Enhancement Tool is referred to as DB2 UET. DB2 UET provides several enhancements for the DB2 utilities, and it can also block and cancel threads on DB2 objects.

These topics provide instructions for installing, configuring, and using DB2 Utilities Enhancement Tool.

These topics are designed to help system programmers, DB2 database administrators (DBAs), production control coordinators, and applications programmers perform these tasks:

- Plan for the installation of DB2 UET
- Customize DB2 UET after installation
- Implement the enhancements for the DB2 utilities
- Block and cancel threads on DB2 objects
- Diagnose and recover from product-related problems

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

- The z/OS® operating system
- ISPF
- Job Control Language (JCL)
- DB2 for z/OS
- Assembler language (if you are coding the user exits that the product supports)

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

http://www.ibm.com/software/data/db2imstools/db2tools-library.html

Chapter 1. Overview

DB2 UET provides several distinct features that can help DB2 data administrators (DBAs), system programmers, and other IT professionals manage their DB2 resources. The product provides enhancements that augment certain DB2 online utilities. The product can also block and cancel threads on DB2 objects. This action enables the objects to be accessed by DB2 utilities or other applications that use DB2 databases.

DB2 UET runs as a started task on a z/OS system. A single product configuration can handle requests from multiple users concurrently and perform product processing across multiple DB2 subsystems on the same z/OS image or in the same data sharing group.

The product provides three interfaces: the DSNUTILB intercept, the ISPF interface, and the batch interface. Your choice of interface depends on the task to be performed.

Topics:

What's new

This topic summarizes the technical changes for this edition.

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

Version 2.2 SC19-3417-05 (sixth edition) - April 2016

Enhancements to processing with the SHRLEVEL REFERENCE and IFDISCARDS extended syntax options for the DB2 LOAD utility are as follows. For more information, see "Loading a table space in RO access mode (SHRLEVEL REFERENCE option)" on page 284 and "Validating records before committing changes (IFDISCARDS option)" on page 281.

- The DB2 UET started task has been enhanced to automatically attach to another member of a data sharing group on the same LPAR when the primary DB2 subsystem goes down. If the primary subsystem is not a member of any data sharing group, the started task does not connect to another DB2 subsystem, and

instead waits until the primary subsystem becomes available. Ensure that the POLICY specifies the DB2 subsystems that will request processing from DB2 UET.

- To prevent RESOURCE UNAVAILABLE errors when two or more LOAD utilities run simultaneously against different partitions of the same partitioned table space, the option DSNUTILB_RETRY_COUNT was added to the options module. Set the value for the new option by specifying a value for **DSNUTILB retry count** on the Tools Customizer Product Parameters panel. The option controls the number of times that DB2 UET retries running the REPAIR utility before timing out.
- When the SHRLEVEL REFERENCE or IFDISCARDS option is specified for the DB2 LOAD utility, during processing the production object is placed in read only (RO) status. However, DB2 still allows DDL to be executed against the production object. When the LOAD utility has completed, DB2 UET checks the table's DDL in the DB2 catalog to verify that the production object did not change while the LOAD utility executed on the shadow object. If DB2 UET detects a change in the table's definition, it stops processing against the shadow object and issues a message to indicate that the production object's definition changed. To support this enhancement, column ALTEREDTS was added to table ABPTABLES in table space ABPPRETB.

  If you enable this enhancement in an existing installation via PTF, complete the following steps to drop and recreate the table space and table with the added column.

  1. Run Tools Customizer.
  2. On the Product Parameters panel, select the following items to generate job ABPOBJAL:
     - Step **Create customized DB2 UET jobs**.
     - Task **Update existing DB2 UET objects**.
  3. Submit job ABPOBJAL.
- DB2 UET provides improved processing of compressed image copies and inline copies to tape. DB2 UET detects that an inline image copy is present within the LOAD utility syntax and removes the syntax for the inline image copy. The LOAD utility runs on the shadow object as normal. DB2 UET then runs a full image copy against the shadow object as a separate utility step. This process enables the image copies to be either allocated to tape or allocated to compressed DASD packs.
- Enhanced table versioning checks are as follows when you specify the SHRLEVEL REFERENCE or IFDISCARDS option:
  - When you specify LOAD REPLACE and
    - All partitions, or the object is nonpartitioned: DB2 UET continues to load the table space and no expanded versioning checks are required.
    - A subset of partitions: DB2 UET performs a check to determine whether the non-loaded partitions were altered since the last reorganization. If some non-loaded partitions were altered but not reorganized, DB2 UET stops processing and issues message `ABPU5556E Partition <partition_number> should be reorganized.` You must reorganize the partitions that are not being loaded.
  - When you specify LOAD RESUME and a subset of partitions, DB2 UET performs a check on all partitions. If partitions were altered since the last reorganization, DB2 UET stops processing and issues message `ABPU5556E Partition <partition_number> should be reorganized.` You must reorganize all partitions.

  - If the versioning requirements described in this section are met, DB2 UET internally accommodates table versioning by performing a versioning ALTER on the shadow object to bring it to the current version.
- The following authorization activities were moved to the started task:
  - Execution of IDCAMS utility (DFSMS AMS). DB2 UET uses the LISTCAT, ALTER, DELETE, DEFINE commands of IDCAMS.
  - Execution of the ADRDSSU utility (DFSMSdss). DB2 UET uses the COPY command of ADRDSSU.
  - Repairing VSAM data sets.
- Options for the DFSMSdss COPY command were added for SHRLEVEL REFERENCE processing:
  - FCTOPPRCP: The value for this option is gathered from ZPARM parameter FLASHCOPY PPRC.
  - FASTREP: The value for this option is gathered from the options module option COPY_FASTREP.
  - DEBUG: The value for this option is gathered from the options module option COPY_DEBUG.

  Message ABPU5552I was added to display the output of the DFSMSdss COPY command.
- DB2 APAR PI08421 provided the DB2 LOAD utility with the parameter BACKOUT YES, which determines the action to be taken if data rows are found in violation during the data validation phase. DB2 UET runs the LOAD REPLACE SHRLEVEL REFERENCE extended functionality on a shadow object, which is a copy of the production object. Since BACKOUT YES is not necessary to run on a shadow object, DB2 UET now detects this parameter and stops the LOAD utility. A message is issued to explain the incompatibility of this new parameter with SHRLEVEL REFERENCE.
- Improved ability to estimate available memory for multiple, simultaneous DFSMSdss copy tasks. This improvement reduces the need for you to manually specify the DSCOPY_LIMIT parameter.

Additional enhancements and processing changes are as follows:

- DB2 UET now detects when a DB2 subsystem starts if the SSID is referenced by a DB2 UET policy. Messages ABPS0607I, ABPS0608W, and ABPS0609I were added for this enhancement.

Version 2.2 SC19-3417-04 (fifth edition) - December 2014

DB2 Utilities Enhancement Tool is a component of the DB2 Utilities Solution Pack. DB2 UET manages a DB2 table, SYSAUTO.UTILITYRUNS_HISTORY, that contains execution data for all utilities that are intercepted by the DB2 UET DSNUTILB intercept. The DB2 UET component of the Solution Pack must be installed and configured for DSNUTILB interception. During customization of the Autonomics Director for DB2 for z/OS component, the SYSAUTO.UTILITYRUNS_HISTORY table is created. DB2 UET inserts rows into the utility execution table as intercepted utilities are run on the system. Topic updates for the solution-only feature are as follows:

- **BBY load library DSN** was added.
- Information about the ACTION attribute value AUTO_DIRECTOR was added.

DB2 UET no longer uses the work database and table space for a dynamically created mapping table. The product now uses an implicit database and table space

for the mapping table. This change removes the potential for -911 SQL code on mapping table creation if another job has a lock on the work table space.

Version 2.2 SC19-3417-03 (fourth edition) - August 2014

DB2 Utilities Enhancement Tool and the IBM® InfoSphere® Change Data Capture (CDC) product can now run on the same system without conflicts.

DB2 UET now detects whether CDC programs CHCADMTR and CHCMIT have run the DSNUTILB utility program. If detected, then the DB2 UET intercept is not performed and control is returned to DSNUTILB for execution to continue. To enable this enhancement, install the product and apply all maintenance.

- Updates to Tools Customizer topics are as follows:
  - The topic "Preparing to customize" was expanded to include an end-to-end customization checklist and planning worksheets to facilitate customization of DB2 UET. The topics formerly located in the Customization Reference were moved into this section.
  - Instructions for configuring DB2 UET in an ACF2 environment were added.
- Enhancements to DB2 UET functionality include the following:
  - Implemented an enhancement to the LOAD utility to prevalidate data. DB2 UET validates records in the SYSREC file against the check constraints and data types of the table that you specify in the LOAD utility syntax. If DB2 UET detects errors in the load, you can either fail the load or pause the load so that you can examine the errors. You can then decide whether to restart the load despite the errors.
  - Implemented support for the LOAD utility to enable LOAD REPLACE SHRLEVEL REFERENCE processing.
  - Implemented support for the date, time, and time stamp formats of the IBM DB2 High Performance Unload for z/OS (HPU) product for LOAD processing.
  - Implemented the ability to substitute the IBM DB2 High Performance Unload for z/OS (HPU) product for the DB2 UNLOAD utility.
- Update to security requirements: The DB2 UET started task must run under a user ID that has SYSADM or SYSCTRL privileges; SYSOPR with MONITOR1 is no longer sufficient.
- Updates to support features of DB2 Version 11 include the following:
  - Expanded RBA/LRSN fields to 10 bytes. In addition to increasing the field length in the ISPF panels, the following examples were updated in this user's guide:
    - Display Threads Detail panel example
    - Threads Canceled Unit of Recovery Report example
    - All Active Threads Unit of Recovery Report example
  - Support for parameters MAPPINGTABLE and MAPPINGDATABASE.

    When running on a DB2 V11 subsystem, parameters MAPPINGTABLE and MAPPINGDATABASE are mutually exclusive. If MAPPINGDATABASE is specified, DB2 UET honors the DB2 V11 parameters, and does not build a mapping table.
  - Support for expanded RBA/LRSN values in mapping tables. After a subsystem has been upgraded to DB2 V11 and you have implemented use of the expanded RBA/LRSN, DB2 UET creates the mapping table with the expanded column width.

Version 2.2 SC19-3417-02 (third edition) - July 2012

- Added MODIFY commands DISPLAY PRACTICE and LIST PRACTICE.
- The DB2 UET Message Monitor now ends with the highest return code that was encountered. In previous versions, it ended with the last return code that was encountered.
- Documented the restriction against using the DISCARDTO keyword in the CHECK DATA utility with clone or auxiliary tables.
- Added message ABPU5916E to indicate that the DB2 UET started task encountered an SQL error.
- The parameter STRIP is supported for PRESORT keys. The STRIP specification is applied to the column in the presort key before presorting. This enhancement ensures that the data is presorted in the correct order. Previously, in LOAD, if a column was part of the PRESORT key and the column also had the STRIP keyword associated with it, the STRIP keyword was either ignored or flagged as an error, depending the type of STRIP that was specified.
- The new JOURNAL_LIMIT attribute was added to the MESSAGE element. JOURNAL_LIMIT enables you to limit the number of duplicate message IDs that get logged in the DB2 UET Utility Monitor journal tables by batch jobs.
- Utility Monitor now conditionally adds parameters based on the presence of a another parameter. The new OPTIONIF attribute was added to the SYNTAX element to specify a text string that must match an identical string in the utility syntax before ADD, REMOVE, or VALUE/SUBSTITUTE actions are performed.
- The SHRLEVEL and UTILITY_COMMAND attributes of the policy <RULE> element no longer have default values.
- Added message ABPS0835I to the SYSPRINT to indicate the current PRACTICE rule in use at DB2 UET startup.
- Added message ABPS0521I to the SYSPRINT to indicate suppression of thread blocker for WORKFILE or TEMP databases.
- Added messages ABP900I through ABP905E to the SYSPRINT to display status and error information about the installation DISCOVER process.

Version 2.2 SC19-3417-01 (second edition) - December 2011

- Implemented a new method of writing to the SYSPRINT DD during DSNUTILB interception. The section about messages in the JESMSGLG data set for the utility was removed.
- Added support for migrating existing Version 2.1 Utility Monitor JOURNAL tables; added the **Create JOURNAL Migration job** (A3MIGRaa) to the Tools Customizer customization jobs. Member names of other jobs changed as a result of the additional job.
- The MESSAGE element was added to the policy PRACTICE description.
- The WSL attribute no longer results in no Message Monitor actions regardless of value, and the restriction was removed from this document.
- The XML element reference information was moved to the appendix.

Version 2.2 SC19-3417-00 (first edition) - August 2011

- Enabled the PRESORT parameter with the LOAD utility for use with Hash Tables. Currently, DB2 UET enables users to presort their SYSREC data prior to the LOAD utility using the parameter PRESORT in the utility syntax. Prior to this enhancement, PRESORT could only be used for objects using an index to sort the table data. After this enhancement, users will be able to presort table data using hash keys for those tables that have been defined as ORGANIZE BY HASH and have CCSID EBCDIC.

- Implemented the Message Monitor. A new MESSAGE element is introduced to the Utility Monitor Policy Rules to define and specify messages to the Message Monitor. Messages are parsed by the DSNUTILB intercept at utility run time as they are written to utility job SYSPRINT.
- Added utilities to the Utility Monitor.
- Implemented support for IBM DB2 Sort. This enhancement enables DB2 UET to use the new IBM sort utility called DB2 Sort for z/OS. When the DB2 utility LOAD is run with the DB2 UET extended syntax option PRESORT, DB2 UET will sort the data through DB2 Sort rather than another sort utility. Added new DSNUTILB intercept messages ABPU5908I, ABPU5909W, ABPU5910I, ABPU5911I, and ABPU5912I.
- Integrated the customization process with IBM Tools Customizer for z/OS (Tools Customizer). The Tools Customizer product is now packaged with DB2 UET to provide users with the capability of customizing DB2 UET using a common user interface.
- Implemented the Utility Monitor. The DB2 UET administrator can specify "shop standards" or "best practices" for running DSNUTILB utilities under control of a DB2 UET policy. The DSNUTILB intercept feature of DB2 UET can monitor the utility syntax and respond in various ways.
- Enabled the Utility Monitor to intercept the RUNSTATS utility.
- Added new DSNUTILB intercept message ABPU5907E. The utility job abends with a S0C4 error if SYSPRINT is absent.
- Added new DSNUTILB intercept messages ABPU5020E and ABPU5021E. STC does not fail initialization when encountering invalid OPTIONS parameters specified for ABPBMAIN_CANCEL_MEMBER or ABPBMAIN_GLOBAL_MEMBER.
- Added new batch interface messages ABPB6800I and ABPB6801I. Added new option to print a Module Entry Point List (MEPL) report to SPRT0000 DD using the ABP batch interface.
- Corrected the description of Utility Monitor ADD attribute.
- Modified batch interface message ABPB6050I and started task message ABPS0822I. Implemented extended DB2 VSN checking.
- Limit duration of REORG utility jobs. Long running SHRLEVEL NONE REORG utility jobs can run past their allotted batch window and sometimes must be canceled by the operations staff. Provided a method by which the utility will terminate itself if a time limit or time of day is exceeded.
- Corrected parsing failure that occurs when encountering CREATOR.TABLE_NAME where CREATOR is greater than 8 bytes.
- CHECK DATA with the Delete option requires exception tables for all checked tables. An option was established to discard the rows to a data set, which can be managed by LISTDEF and TEMPLATE.

What does DB2 UET do?

DB2 UET provides enhancements that are targeted for DB2 LOAD, REORG TABLESPACE, CHECK DATA, and UNLOAD utilities. It also blocks and cancels threads on DB2 objects for these and other DB2 online utilities and any applications and programs in your environment that use a DB2 database.

The DB2 utilities enhancements are as follows:

- Extends the native DB2 LOAD syntax by providing additional options.

- Automatically sizes and creates the mapping table and mapping-table index that are required for the DB2 REORG TABLESPACE utility when the SHRLEVEL CHANGE option is specified. Also automatically drops these objects when reorganization processing completes to preserve space.
- Extends the native CHECK DATA syntax by adding the keyword DISCARDTO, which allows users to discard data to a flat file when running the CHECK DATA utility.
- Substitutes the IBM DB2 High Performance Unload for z/OS product (HPU) for the DB2 UNLOAD utility. When a policy dictates that DB2 UET is to replace DB2 UNLOAD for HPU, then DB2 UET treats the syntax as though it were a normal DB2 UNLOAD. Normal syntax validation occurs, via DSNUTILB, and DB2 UET invokes HPU.

For DB2 online utilities, and for any applications and programs in your environment that use a DB2 database, DB2 UET can transparently cancel active threads on DB2 objects that are needed by DB2 online utilities at runtime. DB2 UET can also block new threads from forming on these objects until after thread-cancelation processing completes.

DB2 UET intercepts the DSNUTILB program based on a user-defined policy to implement the enhancements for the DB2 LOAD and REORG TABLESPACE utilities and to block and cancel threads for these and other DB2 utilities at runtime.

DB2 UET provides an interactive ISPF interface from which you can display a filtered list of active threads, drill down to details about a thread that might be causing access problems, cancel threads selectively, view messages from cancelation processing, and temporarily override some of the started task initialization options.

DB2 UET enables you to create batch job steps for blocking and canceling threads for applications and utilities that you run during the batch window. A single thread-cancelation job step can include many cancel requests, each for a different set of threads.

DB2 UET only supports the syntax structure of the utility commands as they are documented in the *DB2 for z/OS Utility Guide and Reference*.

Features and benefits

DB2 UET provides several distinct features that augment DB2 online utilities and help you manage your DB2 resources more effectively. Learn about the major features that DB2 UET provides and the benefits that you can gain from using them.

DB2 CHECK DATA utility enhancements

New syntax is provided to allow the CHECK DATA utility to write exception rows from checked tables to sequential data sets instead of DB2 tables.

The DB2 UET adds the DISCARDTO and optional DISCARDSPACE keywords as an alternative to the USE table_name2 clause. The DISCARDTO option is used to specify the name of a data file and a control file where discarded rows and a LOAD utility control statement will be written. The files specified in the DISCARDTO syntax can optionally be managed by TEMPLATE statements.

For more information, see Chapter 7, “Writing exception rows to a data set (CHECK DATA utility enhancements),” on page 263.

DB2 LOAD utility enhancements

DB2 UET extends the native DB2 LOAD syntax by providing several additional options that enable you to manipulate the input data before it is loaded, and improve performance. These options are implemented only through the DSNUTILB intercept interface.

The extended LOAD options are:

CONSTANT
Replaces the value for a specific field in the input records for the LOAD utility with another value that you specify. This option is useful for updating or correcting input records before the records are loaded into the database.

VALUEIF
Replaces the value for a specific field in the input records for the LOAD utility with another value that you specify. This replacement occurs only in the input records that match the “field selection criterion” that you specify in the VALUEIF statement. This option is useful for updating or correcting input records before the records are loaded into the database.

PRESORT
Sorts the rows in the input data set by table object identifier (OBID) and by clustering index key (or by the oldest defined index if no clustering index key is available), or by Hash key, before loading the data into the target tables. This option can help improve database performance.

Extended date, time, and timestamp options
Specify as input the external date, time, and timestamp formats that the IBM DB2 High Performance Unload Utility (HPU) provides as output.

IFDISCARDS
Validates records in the SYSREC file against the check constraints and data types of the table that is specified in the LOAD utility syntax.

SHRLEVEL REFERENCE
Enables support for the LOAD utility to add LOAD REPLACE SHRLEVEL REFERENCE processing.

Related concepts:

Chapter 8, “Manipulating data in input records before loading (LOAD utility enhancements),” on page 267
DB2 UET provides several options for the DB2 LOAD utility to enhance load processing. These options are in addition to those that the native DB2 LOAD utility provides. They are implemented only through the use of the DB2 UET DSNUTILB intercept interface.

DB2 REORG TABLESPACE utility enhancement

DB2 UET can automatically size and create the mapping table and mapping-table index that are required for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option. When the reorganization completes, these objects are automatically dropped to preserve space.

This enhancement is implemented only through the use of the DSNUTILB intercept. DB2 UET inserts the DDL for creating the mapping-table objects and the

DDL for dropping the mapping-table objects in the input data set for the REORG TABLESPACE utility. DB2 UET also adds the MAPPINGTABLE option on the SHRLEVEL CHANGE parameter in the input data set. You can control whether to override or preserve any existing MAPPINGTABLE specification that you previously defined.

This enhancement can help you create a mapping-table index that is properly sized and save time in managing mapping-table objects.

For more information, see Chapter 9, "Automatically creating mapping tables (REORG TABLESPACE utility enhancements)," on page 291.

Related concepts:

Chapter 9, "Automatically creating mapping tables (REORG TABLESPACE utility enhancements)," on page 291
DB2 UET can automatically size and create the mapping table and mapping-table index that are required for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option. When the reorganization completes, these objects are automatically dropped to preserve space.

DB2 UNLOAD utility enhancement

DB2 UET can substitute the IBM DB2 High Performance Unload for z/OS (HPU) product for the DB2 UNLOAD utility.

DB2 UET provides the policy attribute <REPLACE> to the element <UTILITY> to enable this enhancement.

For more information, see Chapter 10, "Substituting HPU for the DB2 UNLOAD utility," on page 295.

Thread cancelation

DB2 UET enables you to block and cancel threads on DB2 objects that are needed by DB2 online utilities at runtime or by applications and programs that run during the batch window.

From the ISPF interface, you can selectively cancel threads that might be causing DB2 access problems.

Using the DSNUTILB intercept or batch interface, you can block new threads as well as cancel existing threads to provide utilities or applications with exclusive access to DB2 objects. You can also perform trial runs of thread cancelations so that you can check that the correct threads will be canceled before actually implementing thread-cancelation processing in your production environment.

For more information, guidelines for determining which interface to use, and restrictions, see Chapter 5, "Blocking and canceling DB2 threads," on page 173.

Related concepts:

Chapter 5, "Blocking and canceling DB2 threads," on page 173
You can cancel active DB2 threads by using any of the DB2 UET interfaces. By canceling threads, you can free DB2 objects that are being held so that other applications or utilities can access them.

DSNUTILB interception

DB2 UET can intercept the DB2 DSNUTILB program based on a user-defined policy to implement the enhancements for the DB2 LOAD and REORG TABLESPACE utilities and to transparently block and cancel threads for these and other DB2 utilities.

For interception to occur, you must create a single intercept policy in XML and specify the name of the policy member in the started task PROC. The policy defines all intercept processing that you want the started task to perform. The policy must at least specify the DB2 subsystem or subsystems on which to perform intercept processing. The group attach name is not a valid entry for the policy. For thread blocking and cancelation, the policy must also include rules for selecting the utilities, users, or objects for which to block and cancel threads.

To implement the DB2 UET options for the DB2 LOAD utility, you must manually add the options to the LOAD utility JCL. However, to implement the mapping-table enhancement for the REORG TABLESPACE utility or to block and cancel threads, *no changes to the existing utility JCL are required*.

After you complete the necessary configuration tasks, the started task can intercept DSNUTILB to transparently implement the enhanced utility processing that you configured whenever the utilities are run by DSNUTILB. For thread blocking and cancelation, the DSNUTILB intercept analyzes the intercept policy and the DSNUTILB SYSIN stream for a utility to determine which threads to block and cancel.

DB2 UET supports the standard DB2 restart of a DB2 utility for which interception is occurring or has occurred. No special user intervention is required.

For more information, see “Preparing to intercept the DSNUTILB program and DB2 utilities” on page 122.

Related concepts:

“Preparing to intercept the DSNUTILB program and DB2 utilities” on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

Maintenance utility

DB2 UET provides a maintenance utility (ABPMAINT) that you can use to perform special maintenance tasks that are related to DSNUTILB interception.

You should use the ABPMAINT utility to perform these tasks:

- Terminate a DB2 utility for which DSNUTILB interception is occurring or has occurred. The ABPMAINT utility both terminates the DB2 utility and removes any intercept data that is associated with the utility ID from the appropriate DB2 tables.
- Indicate the point at which a DB2 utility should resume its intercept-enhanced processing after the DB2 utility ends because of an exceptional circumstance, such an abend of DB2, which makes a normal DB2 restart not possible.

For information about using the ABPMAINT utility, see “Restarting a DB2 utility in exceptional circumstances” on page 319 and “Terminating a DB2 utility for which interception has occurred” on page 318.

Related tasks:

“Restarting a DB2 utility in exceptional circumstances” on page 319
When a DB2 utility for which DSNUTILB interception is occurring terminates abnormally, DB2 can usually resume utility processing from the appropriate point, without any special user intervention, when you restart the utility. However, in some exceptional circumstances, a normal DB2 restart of a DB2 utility might not be possible. In these circumstances, you can use the DB2 UET ABPMAINT utility to resume utility processing.

“Terminating a DB2 utility for which interception has occurred” on page 318
If you need to terminate a DB2 utility for which DSNUTILB intercept processing is occurring or has occurred, use the ABPMAINT utility that is provided by DB2 UET.

User exits

DB2 UET supports optional user exits for controlling user access to certain product functions and ISPF panels or for performing any additional processing that you determine is necessary before or after thread cancelations.

You can create the following types of user exits:

- A *security exit* for controlling user access to product functions (for example, the cancel commands) and to certain ISPF panels. If you do not create a security exit, all users will be able to access all product functions.
- A *pre-cancel exit* to perform any additional processing that you need to perform prior to thread cancelations. For example, you might want to prevent threads that are running under a particular plan from being canceled.
- A *post-cancel exit* to perform any additional processing that you need to perform after thread cancelations. For example, you might want to notify users that thread cancelation processing occurred.

DB2 UET provides sample exits and DSECTs in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specify during product customization. You can edit the sample exits, as needed, but you cannot change the DSECTs.

As delivered in DB2 UET V2.1 base code, all three exits exist in SABPLOAD and are only stubs; that is, no functionality performed as delivered. To *turn on* this functionality, the SABPSAMP code member(s) must be modified, assembled and link edited. The newly-modified load modules must be either overwritten in SABPLOAD or written to a new load library. If written to a new load library, then:

1. The new load library must be APF-authorized.
2. The new load library must be concatenated in *front* of SABPLOAD in the STC STEPLIB.

   Note: If concatenated *behind* SABPLOAD, the new or modified user exit(s) will not be invoked.

The ASSEMBLY step should contain the SABPSAMP library added to the SYSLIB DD. The SABPSAMP library contains some DSECTs that is needed by the ASSEMBLY step.

For more information about these user exits, see “Security exit” on page 451, “Pre-cancel exit” on page 456, and “Post-cancel exit” on page 462.

Related reference:

“User exit details” on page 451
These topics provide reference information for each type of DB2 UET user exit that you can optionally define: a security exit, a pre-cancel exit, and a post-cancel exit. This information is primarily intended for the assembler-language programmers who will develop the exits.

Auditing and logging features

DB2 UET logs product messages and stores audit information in DB2 tables. The product administrator can use this information to troubleshoot problems and audit product activities.

During product configuration, the product administrator determines whether audit and logging data are recorded for the DB2 UET configuration by setting the appropriate started task initialization options. The administrator can temporarily enable or disable the recording of audit or logging data from the Control System panel in the ISPF interface.

When the audit status is active, DB2 UET records information about product activities in the ABPAUDIT table. You can query this table to assess product activities. For example, you can determine which threads were canceled, who canceled the threads, and when the cancelations occurred.

When the logging status is active, DB2 UET records all messages from product operations in the ABPLOG table. You can use this information to diagnose problems and optimize product performance.

Related concepts:

“Obtaining information from the audit and logging tables” on page 308
You can create SQL queries to obtain information from the audit and logging tables for audit or troubleshooting purposes. For example, you could create a query to identify the users who were canceling threads during a certain period or to gather messages related to a specific operation for diagnostic use.

Interfaces and components

DB2 UET includes several key components, including three alternative interfaces. Learn about these components to prepare for product customization and use.

The key components are:

- IBM Tools Customizer for z/OS, which provides a single process for customizing products.
- The started task, which communicates with DB2 to perform product functions.
- The DSNUTILB intercept interface, which implements the enhancements for the DB2 utilities and transparently blocks and cancels threads on DB2 objects for DB2 utilities at runtime.
- The ISPF interface, which you can use to display information about active DB2 threads, cancel threads, and perform product administration tasks.
- The batch interface, which you can use to block and cancel threads on DB2 objects that are needed by applications and programs that you run as part of a batch job.

- The supervisor call (SVC), which enables communication between the product interfaces and the started task.

Related concepts:

"Sample product configuration" on page 18
A sample configuration of DB2 UET on a single z/OS image is presented.

Tools Customizer overview

IBM Tools Customizer for z/OS (also referred to as Tools Customizer) standardizes many of the customization processes that are required to customize IBM Tools that run on z/OS. Tools Customizer is a component of IBM Tools Base for z/OS.

Tools Customizer provides a consistent ISPF interface to ensure that the customization process is the same for all IBM Tools products and solution pack components. It also provides the ability to "discover" parameter values from products or solution pack components that you previously customized manually or by using Tools Customizer.

Features and benefits

Tools Customizer provides the following features:

- A single, consistent ISPF interface ensures that the customization process is the same for all IBM Tools products and solution pack components.
- A Discover EXEC discovers values for common product and DB2 parameters from a product or solution pack component that you previously customized manually or by using Tools Customizer. Each IBM Tools product and solution pack component has a unique Discover EXEC. The discovered parameters are stored in the data store. If the product or solution pack component that you want to customize exists in the Tools Customizer data store, Tools Customizer issues a warning before it overwrites existing values. Use the Discover EXEC by issuing the DISCOVER command on the Customizer Workplace panel.
- The data store retains discovered and manually specified parameter values. Because the parameter information is persistently stored, you have to manually specify or discover parameter values only once. Tools Customizer uses these parameter values where they are applicable.
- A metadata repository contains the members that define the following customization attributes for products and solution pack components:
  - Parameters, tasks, and steps for the product or solution pack component to be customized. Some product or solution pack parameters, tasks, and steps are required.
  - DB2 parameters for the DB2 subsystem, DB2 group attach name, or DB2 data sharing member on which you will customize the product or solution pack component. All of the DB2 parameters are required.
- Default values are provided for product parameters and solution pack component parameters and DB2 parameters. The default values show examples of how to complete fields.

Started task

The DB2 UET started task is a program that remains active and ready to perform work on behalf of the product interfaces.

The started task receives input from the interfaces through the SVC and then communicates with the DB2 subsystems to implement the supported DB2 utility

enhancements or to block and cancel threads. A single started task can process simultaneous requests from multiple users across the system.

During customization, you must set several options for the started task in the initialization options member *abpid*OPTS, where *abpid* is the configuration ID that you specify at customization. For example, you must set the option that specifies the primary DB2 subsystem where the audit and logging tables reside.

After you start the started task, you can perform product functions by using any of the product interfaces.

Related concepts:

"Considerations for running multiple started tasks" on page 32
A single started task is usually sufficient to handle multiple user requests from any of the DB2 UET interfaces to perform work on one or more DB2 subsystems. However, if you have a very high volume of activity, you can run multiple started tasks concurrently to handle the workload more efficiently.

Related tasks:

"Setting up additional initialization options members (optional)" on page 97
The initialization options member, called *abpid*OPTS, is used by the DB2 UET started task upon initialization. Tools Customizer customized *abpid*OPTS for the ABPID that you specified during customization. If you plan to run multiple started tasks to monitor different DB2 subsystems, you must create separate initialization options members for each. This customization step is required if you will use multiple started tasks.

"Starting the started task" on page 109
Start the DB2 UET started task so that you can begin using the product interfaces to block and cancel threads or implement the DB2 utility enhancements. This customization step is required.

DSNUTILB intercept interface

The DB2 UET DSNUTILB intercept interface intercepts the DB2 DSNUTILB program to implement enhancements that are designed specifically for the DB2 LOAD and REORG TABLESPACE utilities, and to block and cancel threads on DB2 objects for these and other DB2 utilities.

You must use the DSNUTILB intercept to implement the following utility-specific enhancements:

- The additional options that DB2 UET supplies for the DB2 LOAD utility
- The automatic creation of the mapping table and mapping-table index that are required for the DB2 REORG TABLESPACE utility when the SHRLEVEL CHANGE option is specified, and the automatic removal of these objects after the REORG TABLESPACE utility completes

You can optionally use the DSNUTILB intercept to block and cancel threads on DB2 objects that are needed by DB2 utilities at runtime. The DSNUTILB intercept cancels active threads and blocks new threads from forming on the DB2 objects just before a utility needs to access the objects. No changes to the utility JCL are required. Alternatively, you could cancel threads for DB2 utilities from the ISPF or batch interface; however, the DSNUTILB intercept has the advantage of intercepting DSNUTILB to transparently perform thread blocking and cancelation whenever DSNUTILB invokes the utilities.

To use the DSNUTILB intercept to perform the supported tasks, you must create an intercept policy in XML. The policy identifies the DB2 subsystems on which to

perform intercept processing and defines any rules that you want to use for selecting the DB2 utilities, users, and objects for which to block and cancel threads.

Related concepts:

"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

ISPF interface

DB2 UET provides an interactive ISPF interface from which you can easily assess the status of active threads and cancel any threads that are preventing access to DB2 resources.

From the ISPF interface, you can perform the following tasks:

- View a list of all active threads, including their primary attributes
- Filter the list of threads based on numerous thread-filtering criteria
- View detailed information about a single thread
- Display the DB2 databases and page sets that a thread is referencing
- Cancel a single thread or multiple threads by using either the DB2 -CANCEL THREAD command or the z/OS Cancel command
- View DB2 and DB2 UET messages from the cancelation processing of a thread

If you are the product administrator, you can also monitor any DB2 UET configuration from the ISPF interface. You can view information about the started task and look up the name of the DB2 plan or the initialization options member. Also, you can view the settings for all of the initialization options and temporarily change some of these settings (for example, the option to enable or disable logging), if necessary.

Related concepts:

"Using the ISPF interface" on page 110
Learn about using the DB2 UET ISPF interface.

Related tasks:

"Starting the ISPF interface" on page 110
Start the ISPF interface so that you can use it to view or cancel DB2 threads, view the product status, or override selected started task initialization options.

Batch interface

DB2 UET provides a batch interface that enables you to run thread-cancelation job steps as part of your batch jobs.

You can perform the following tasks using the batch interface:

- Create a job step that contains multiple CANCEL_THREADS requests (also called *cancel requests*), each with a different set of thread-filtering criteria.
- Block new threads from forming on the DB2 objects that a utility or application needs to access until after cancelation processing completes and the utility runs.
- Run a thread-cancelation job in simulation mode to check that the correct threads will be canceled when you actually run the job, or run the job in a mode that checks the parameter syntax only.

- View detailed reports that DB2 UET generates for each CANCEL_THREADS request during an actual or simulated run of a thread-cancelation job.
- Create a thread-cancelation job that uses DB2 NOBACKOUT processing, if necessary, instead of the default BACKOUT processing. (This type of thread cancelation should be performed only in exceptional circumstances because it can result in data integrity problems.)

You can use the return code from a thread-cancelation job step for an application to control whether subsequent applications in the batch job will run.

The batch interface is called by the DSNUTILB interface to perform any intercept processing that you configured in the intercept policy for DB2 utilities.

Related concepts:

"Using the batch interface" on page 119
With the batch interface, you can manually create a job step for canceling threads and include it within a batch job.

Supervisor call (SVC)

The SVC enables the product interfaces to communicate with the DB2 UET started task.

One SVC is required for each started task. You specify the SVC number in the started task initialization options member during customization.

When you start the started task, the specified SVC is dynamically installed. When you stop the started task, the SVC is dynamically removed. No IPL or SYS1.PARMLIB changes are required.

DB2 environment

DB2 UET runs as a started task on a z/OS system. The started task communicates with DB2 to perform product functions and to store information about product activities in DB2 tables.

You can run a single DB2 UET started task to perform intercept processing and cancel threads on multiple DB2 subsystems. The started task communicates with one DB2 subsystem, called the *primary subsystem*, to record audit and logging information about product activities in DB2 tables. This subsystem is identified by the DB2_SSID option in the started task initialization options member. The started task can also communicate with other DB2 subsystems that you define.

During customization, you identify the primary subsystem and each *additional subsystem* that you want to use. Based on your input, the Tools Customizer creates customized batch jobs for creating the necessary DB2 objects on each of these subsystems. When you run the batch jobs, the following tables are created:

- On the primary subsystem: the tables for audit information, logged messages, thread-blocking data, and DSNUTILB-intercept-worklist data.
- On each additional subsystem: only the tables for thread-blocking and DSNUTILB-intercept-worklist data

Tip: In DB2 data sharing environments, all subsystems in a data sharing group share the same DB2 catalog. Consequently, you can create the thread-blocking and worklist tables on any single member within the group.

The Tools Customizer also tailors the started task initialization options member based on your input. This member includes options that 1) specify the primary subsystem and the DB2 DSNLOAD library, 2) control DB2 connections, and 3) control DB2 tasks. You can edit the options member, if necessary.

Important: If you plan to monitor threads across multiple DB2 subsystems that have different DB2 versions, ensure that you specify the earliest (lowest) of these DB2 versions as the DSNLOAD library. This DSNLOAD library must appear in the STEPLIB concatenation of the DB2 UET started task PROC. Otherwise, connection problems might occur when you attempt to manage threads on DB2 subsystems other than the primary subsystem.

When you use any DB2 UET interface to perform a product function, you must specify the DB2 subsystem on or from which you want to perform the function. (This subsystem can be the primary subsystem or one of the additional subsystems that you defined at customization.) You specify this subsystem from the product interfaces, as follows:

- In the DSNUTILB intercept interface: Use the <DB2SYSTEM> element in the DSNUTILB intercept policy.
- In the ISPF interface: Select the subsystem on the Set DB2 System panel.
- In the batch interface: Use the DB2SSID global parameter.

In a non-data sharing environment, if you use the DSNUTILB intercept or batch interface, you will be able to perform product functions only on the DB2 subsystems that you explicitly select. In an intercept policy, you can specify the <DB2SYSTEM> element multiple times to select multiple DB2 subsystems on which to perform intercept processing. However, in a batch job, you can specify the DB2SSID global parameter only once to select a single DB2 subsystem on which to block and cancel threads. If you use the ISPF interface in a non-data sharing environment, you will be able to cancel threads on the DB2 system that you select on the Set DB2 System panel, and after a connection to that DB2 system is established, you will be able to also cancel threads on additional subsystems. The additional subsystems must meet all of these conditions: 1) they are on the same z/OS image as the selected subsystem; 2) they were defined at customization as either the primary subsystem or an additional subsystem; 3) they have a DB2 version that DB2 UET supports; and 4) they have a DB2 plan that has been bound for DB2 UET.

In a DB2 data sharing environment, the behavior is somewhat different. If you use the batch interface or ISPF interface, a single started task configuration can cancel threads on any active local or remote subsystem in the same data sharing group as the primary subsystem. If you use the DSNUTILB intercept, DB2 UET can intercept DSNUTILB only for the DB2 subsystems in the data sharing group that 1) are specified by a <DB2SYSTEM> element within the DSNUTILB intercept policy and 2) are on a z/OS image where a DB2 UET started task configuration is running and an intercept policy is defined. The DSNUTILB intercept can implement the enhancements for the LOAD and REORG TABLESPACE utilities only on these subsystems. However, for thread blocking and cancelation, the DSNUTILB intercept calls the batch interface. Consequently, threads can be blocked and canceled for DB2 utilities that are running against any DB2 subsystem that is in the same data sharing group as the selected subsystem, even subsystems on another z/OS image. For more information, see “Considerations for DB2 data sharing environments” on page 28.

Related concepts:

"Considerations for DB2 data sharing environments" on page 28
Before you deploy DB2 UET in a DB2 data sharing environment, review information about deployment and configuration issues.

"Sample product configuration"
A sample configuration of DB2 UET on a single z/OS image is presented.

Sample product configuration

A sample configuration of DB2 UET on a single z/OS image is presented.

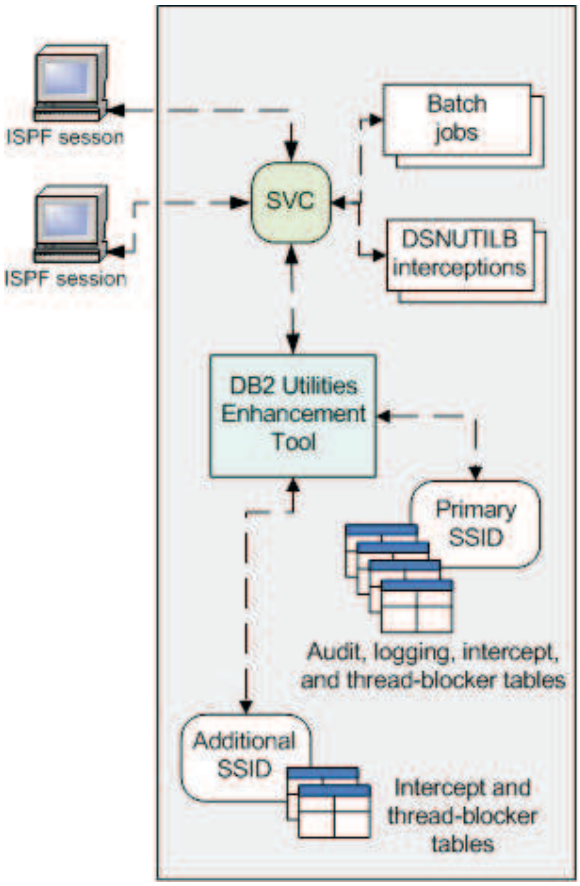


Figure 1. Sample product configuration on a single z/OS image

In this simple configuration, a single DB2 UET started task is processing requests from two ISPF interface users and several batch thread-cancelation jobs and DSNUTILB intercept jobs. The started task is connected to two DB2 subsystems: the *primary subsystem* and an *additional subsystem*. Both of these subsystems were defined at customization. The primary subsystem is specified by DB2_SSID option in the started task initialization options member.

The primary subsystem contains the audit and logging tables for the product. The additional subsystem was selected by a product user as the subsystem to which to connect to perform a product task. Users select this subsystem from the product interface that they are using, that is, from the Set DB2 System panel in the ISPF interface, in the DB2SSID global parameter of a batch job, or in the <DB2SYSTEM> element of the DSNUTILB intercept policy. (Note the product user could have selected the primary subsystem instead.) In a non-DB2 data sharing environment,

each subsystem must contain the tables that store data for thread-blocking and DSNUTILB intercept operations on that subsystem; these tables must be present even if users do not actually perform thread blocking or DB2 intercept processing on the subsystem.

When a user initiates an action from a product interface, the interface calls the SVC to communicate with the started task. The started task then connects to the selected DB2 subsystem to perform product tasks such as canceling threads.

Many other, more complex product configurations are possible.

Related concepts:

"Interfaces and components" on page 12
DB2 UET includes several key components, including three alternative interfaces. Learn about these components to prepare for product customization and use.

"DB2 environment" on page 16
DB2 UET runs as a started task on a z/OS system. The started task communicates with DB2 to perform product functions and to store information about product activities in DB2 tables.

Choosing the interface to use

DB2 UET provides three interfaces: the DSNUTILB intercept, the batch interface, and the ISPF interface. Your choice of interface will depend on the product function that you want to perform.

If you want to implement the DB2 UET enhancements for the LOAD utility or REORG TABLESPACE utility, you must use the DSNUTILB intercept. If you want to cancel threads, you can use any of the interfaces, but one of the interfaces might be more appropriate for your situation. If you want to block threads as well as cancel threads, you must use either the batch interface or the DSNUTILB intercept.

The following table presents some general guidelines for choosing an interface. The left column shows some common tasks that database administrators and system administrators perform, and the right-hand column shows the DB2 UET interface that is best suited for performing each of these tasks.

Table 1. Choosing an interface

To perform this task...	**Use this interface...**
Block and cancel threads on DB2 objects for DB2 utilities that are run by the DSNUTILB program	DSNUTILB intercept
Block and cancel threads on DB2 objects for DBA activities, applications, and utilities that are part of batch jobs that run during the batch window	Batch interface
Cancel active threads selectively to resolve DB2 access problems, or to free DB2 objects that are needed by an application that is run on an ad hoc basis	ISPF interface
Implement the additional keywords that DB2 UET supports for the LOAD utility	DSNUTILB intercept only
Automatically create a mapping table and mapping-table index for the REORG TABLESPACE utility when the SHRLEVEL CHANGE option is specified, and automatically drop these objects when reorganization processing completes	DSNUTILB intercept only

Service updates and support information

Service updates and support information for this product, including software fix packs, PTFs, frequently asked questions (FAQs), technical notes, troubleshooting information, and downloads, are available from the web.

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

http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_Tools_for_z~OS

Product documentation and updates

DB2 Tools information is available at multiple places on the web. You can receive updates to DB2 Tools information automatically by registering with the IBM My Notifications service.

Information on the web

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

http://www.ibm.com/software/data/db2imstools/db2tools-library.html

You can also access documentation for many DB2 Tools from IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter

Search for a specific DB2 Tool product or browse the **Information Management** > **DB2 for z/OS family**.

IBM Redbooks® publications that cover DB2 Tools are available from the following web page:

http://www.redbooks.ibm.com

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

http://www.ibm.com/software/data/db2imstools/solutions/index.html

Receiving documentation updates automatically

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

To register with the My Notifications service:

1. Go to http://www.ibm.com/support/mysupport
2. Enter your IBM ID and password, or create one by clicking **register now**.

3. When the My Notifications page is displayed, click **Subscribe** to select those products that you want to receive information updates about. The DB2 Tools option is located under **Software** > **Information Management**.
4. Click **Continue** to specify the types of updates that you want to receive.
5. Click **Submit** to save your profile.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other IBM product documentation, use one of the following options:

- Use the online reader comment form, which is located at http://www.ibm.com/software/data/rcf/.
- Send your comments by email to comments@us.ibm.com. Include the name of the book, the part number of the book, the version of the product that you are using, and, if applicable, the specific location of the text you are commenting on, for example, a page number or table number.

Accessibility features

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

The major accessibility features in this product enable users to perform the following activities:

- Use assistive technologies such as screen readers and screen magnifier software. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.
- Customize display attributes such as color, contrast, and font size.
- Operate specific or equivalent features by using only the keyboard. Refer to the following publications for information about accessing ISPF interfaces:
  - *z/OS ISPF User's Guide, Volume 1*
  - *z/OS TSO/E Primer*
  - *z/OS TSO/E User's Guide*

  These guides describe how to use the ISPF interface, including the use of keyboard shortcuts or function keys (PF keys), include the default settings for the PF keys, and explain how to modify their functions.

Chapter 2. Preparing to customize

Before you start to customize DB2 Utilities Enhancement Tool for the first time, determine all of the customization values that you need to specify during the customization process, and familiarize yourself with all of the customization tasks.

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

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

Task	Link to detailed instructions	Status
Tools Customizer basics		
Prior to beginning the customization process, familiarize yourself with Tools Customizer terminology and data sets, and other basic information about Tools Customizer.	"Tools Customizer reference" on page 441	
Hardware, software, and storage requirements		
Verify that your environment meets the minimum hardware requirements.	"Verify that your environment meets hardware requirements" on page 26	
Verify that your environment meets the minimum software requirements. To install and use DB2 Utilities Enhancement Tool, your environment must be running a supported version of the z/OS operating system and of DB2 for z/OS. Additionally, certain levels of maintenance might be required.	"Verify that your environment meets software requirements" on page 26	
Verify that your environment meets CSA or ECSA minimum storage requirements, that your DFSMS rules support LOAD utility enhancements, and that your DASD storage is sufficient.	"Verify that your environment meets storage requirements" on page 27	
SMP/E installation		
Verify that DB2 Utilities Enhancement Tool has been installed correctly. DB2 Utilities Enhancement Tool is installed by using standard SMP/E processing.	"Verify that DB2 Utilities Enhancement Tool has been installed successfully" on page 28	
Verify that Tools Customizer for z/OS has been installed correctly. Tools Customizer for z/OS is installed by using standard SMP/E processing.	"Verify that Tools Customizer has been installed successfully" on page 28	
Security requirements		
Make sure that you have the required authorizations to use DB2 UET.	"Security requirements" on page 31	
Started task considerations		
If you have a very high volume of activity, you can run multiple started tasks concurrently to handle the workload more efficiently.	"Considerations for running multiple started tasks" on page 32	
Data sharing considerations		

Task	Link to detailed instructions	Status
Review deployment and configuration issues for DB2 data sharing environments.	"Considerations for DB2 data sharing environments" on page 28	
Dispatching priority requirements		
Ensure that the dispatching priority for the DB2 UET started task is set correctly with respect to other dispatching priorities.	"Dispatching priority" on page 33	
WTO messages for automated operations		
Consider whether to use the Write-to-Operator (WTO) messages that DB2 UET issues for automated operations.	"Use of WTO messages for automated operations" on page 34	
Vector table issue		
DB2 UET uses the shared 4-KB vector table. Review information about shared storage.	"Operational issue related to the vector table" on page 34	
DB2 version migration		
Review actions that you need to take when a DB2 subsystem that is used with DB2 UET is migrated to a later version of DB2, or when a fallback to an earlier version is required.	"DB2 version migration considerations" on page 34	
Gather data set names		
During the customization process, you must specify data set names for the following things: • Tools Customizer • FEC (common code) • DB2 Utilities Enhancement Tool	"Worksheets: Gathering required data set names" on page 35	
Gather parameter values		
During the customization process, you must specify parameter values for DB2 Utilities Enhancement Tool, for DB2, and for your LPAR.	"Worksheets: Gathering parameter values for Tools Customizer" on page 38	
Customize DB2 Utilities Enhancement Tool		
Start Tools Customizer by running a REXX EXEC from the ISPF Command Shell panel.	"Starting Tools Customizer" on page 65	
Set up Tools Customizer user settings. If you are running Tools Customizer for the first time, you must modify several user settings to suit your environment. Otherwise, if the user settings that you have already established are still appropriate, skip this step.	"Modifying Tools Customizer user settings" on page 66	
Complete the steps in the appropriate customization roadmap based on the type of customization that you are performing.		
Customizing for the first time To customize the product for the first time, follow **Roadmap: Customizing DB2 Utilities Enhancement Tool for the first time**.	"Roadmap: Customizing DB2 UET for the first time" on page 72	

Task	Link to detailed instructions	Status
Customizing a different version of the product If you have already customized a version of the product and you want to use the same parameter values to customize a different version, follow **Roadmap: Customizing a new version of DB2 Utilities Enhancement Tool**.	"Roadmap: Customizing a new version of DB2 UET from a previous customization" on page 72	
Recustomizing the same version of the product If you have customized this version of the product for the first time, want to go back and change parameters and regenerate jobs, follow **Roadmap: Recustomizing DB2 Utilities Enhancement Tool**.	"Roadmap: Recustomizing DB2 UET" on page 73	
Some customization options require you to manually complete additional tasks after you have used Tools Customizer. If you generated jobs in Tools Customizer that correspond to the following customization options, complete the additional tasks before you submit the jobs. In some cases, an optional task can be completed either by using Tools Customizer or by manually completing tasks without using Tools Customizer.		
APF authorization		
The SABPLOAD data set must be APF authorized.	"APF-authorizing the load library" on page 91	
Make the started task address space available to user interfaces		
Copy the DB2 UET started task PROC to your system PROCLIB to make the started task address space available to the user interfaces for the product.	"Copying the started task PROC" on page 91	
Copy the DSNUTILF module		
The DSNUTILF module must be in an APF-authorized library in the STEPLIB or JOBLIB concatenation for the DB2 LOAD utility jobs.	"Copying the DSNUTILF module" on page 92	
Optional: Customize DSNUTILB intercept parameters		
You can customize the cancel parameters and global parameters that the DSNUTILB intercept uses for all thread blocking and cancelation activities.	"Customizing DSNUTILB intercept parameters (optional)" on page 93	
Optional: Set up the initialization options member		
The initialization options member, called *abpid*OPTS, is used by the DB2 UET started task upon initialization.	"Setting up additional initialization options members (optional)" on page 97	
Optional: Create a security exit		
You can create a security user exit to control user access to product commands for intercept processing and to certain ISPF panels.	"Creating a security exit (optional)" on page 106	
Optional: Create a pre- or post-cancel exit		
You can create pre-cancel and post-cancel exits to perform additional processing before or after thread cancelations.	"Creating a pre- or post-cancel exit (optional)" on page 107	
Optional: Migrate versions of DB2 Utilities Enhancement Tool data		
You can migrate data from existing DB2 UET AUDIT and LOG tables and JOURNAL tables.	"Migrating existing data" on page 108	
Prepare DB2 UET for use		

Task	Link to detailed instructions	Status
Before you can use the product, you must perform initial set-up tasks.	Chapter 4, “Getting started,” on page 109	

Set up your environment prior to customization

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

Verify that your environment meets hardware requirements

DB2 Utilities Enhancement Tool can be used on any IBM mainframe computer that is capable of running the required software.

Verify that your environment meets software requirements

Ensure that you are using z/OS V1.13 (5694-A01) or later.

Ensure that you are using one of the following supported versions of DB2 for z/OS and appropriate maintenance is applied:

DB2 V8

This is the minimum version that supports basic DB2 UET processing.

- DB2 V8 (5625-DB2) (new function mode)
- DB2 Value Unit Edition V8.1 (5697-N29)

DB2 V9

- DB2 V9 (5635-DB2)
- DB2 Value Unit Edition V9.1 (5697-P12)

DB2 V10

This is the minimum version that supports all DB2 UET features.

- DB2 V10 (5605-DB2)
- DB2 Value Unit Edition V10.1 (5697-P31)

Required maintenance:

- PM93789
- UI28994
- UI30114

DB2 V11

- DB2 V11 (5615-DB2)
- DB2 Value Unit Edition V11.1 (5697-P43)

Required maintenance:

- UI28995
- UI30115

Ensure that you are using a supported version of the following software:

- ISPF V4 (5655-042) or later
- IBM SMP/E for z/OS V03.05.00 (5655-G44) or later
- DB2 Utilities Suite for z/OS V10.01.00 (5655-V41) or later
- IBM Tools Base for z/OS V01.04.00 (5655-V93) or later

(*HPU users only*) To substitute the IBM DB2 High Performance Unload for z/OS (HPU) for the DB2 UNLOAD utility, HPU must be installed on the DB2 subsystem on which the UNLOAD is to be executed.

To use features that prevalidate data before loading it (IFDISCARDS and SHRLEVEL REFERENCE options):

- Ensure that your IBM Workload Manager for z/OS (WLM) environment is set up so that DB2 UET can run the SYSPROC.DSNUTILU stored procedure.
- If the DB2 UET load modules are in the WLM STEPLIB concatenation, or if the DB2 Analytics Accelerator Loader product is installed, ensure that DB2 UET PTF UI17668 is applied and that you are using the modules that are provided with it. Specifically, verify that the compile date in the DSNUTILF module in the WLM address space is the same as the date of the module in the enhancement load library.

To use the SHRLEVEL REFERENCE extended syntax option, ensure that you have applied PTF UI32698.

Verify that your environment meets storage requirements

For each z/OS system on which DB2 UET will run, ensure that the common service area (CSA) or extended common service area (ECSA) meets the minimum storage requirements, that your DFSMS rules support the LOAD utility enhancements, and that your DASD storage is sufficient.

- The CSA and ECSA storage requirements are as follows. The total amount of CSA/ECSA storage that is required will depend on the number of started tasks that you run, the number of users, the level of user request activity, and the number of DB2 subsystems that are defined on the z/OS system.
  - 4 KB from subpool 228 in ECSA for the vector table
  - 1 KB from subpool 241 in CSA or ECSA for each active DB2 UET started task
  - 160 bytes from subpool 241 in CSA or ECSA for each simultaneous request that is made from a DB2 UET interface
  - 44 bytes from subpool 241 in CSA or ECSA for each DB2 subsystem that is defined on the z/OS system
  - DB2 UET supports the date, time, and timestamp values of the IBM DB2 High Performance Unload for z/OS (HPU) product. To use this feature, a minimum amount of storage will be required in CSA/ECSA. All other storage is allocated from private address spaces or data spaces.
- DFSMSdss requirements are as follows:

  When performing data prevalidation for the DB2 LOAD utility and when running the LOAD utility with the parameters REPLACE SHRLEVEL REFERENCE, DB2 UET uses DFSMSdss to copy data sets from the production objects to the shadow objects. If SMS is active, you must set up SMS rules to allow the shadow data sets to reside on volumes that are allowed for DB2 objects. The underlying data sets for the shadow objects will be renamed to become the active page set for the DB2 table and index spaces; they will not be moved from the location in which SMS places them when the shadow objects are created.

The renaming or DFSMSdss COPY of DB2 VSAM objects follows the standard naming conventions for such objects:

catname.DSNDB*x*.*dbname*.*psname*.*y*0001.*znnn*

where y is a letter in the range A through Z. DB2 UET checks for existing data set names and uses the first available letter to rename the objects.

- For DASD storage requirements, see the *Program Directory for DB2 UET for z/OS*.

Verify that DB2 Utilities Enhancement Tool has been installed successfully

See the Program Directory for IBM DB2 Utilities Enhancement Tool for z/OS, GI10-8981 for installation instructions.

Verify that Tools Customizer has been installed successfully

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

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

Considerations for DB2 data sharing environments

Before you deploy DB2 UET in a DB2 data sharing environment, review information about deployment and configuration issues.

A DB2 data sharing group is composed of one or more DB2 subsystems that are located on the same z/OS image or on different z/OS images. The member subsystems share a common DB2 catalog and can directly access and change the same data while maintaining data integrity.

A DB2 UET started task can block and cancel threads and perform DSNUTILB intercept processing on the active subsystems within a data sharing group that have a DB2 version that DB2 UET supports. During customization, you must define at least one member subsystem as the *primary subsystem*. This subsystem must contain the DB2 UET audit and logging tables.

Because all subsystems in a data sharing group share the same DB2 catalog, they can also share the same thread-blocker and DSNUTILB intercept tables and (for the REORG TABLESPACE mapping-table enhancement) the same mapping-table database and table space. You can define these objects once on any active member subsystem in the data sharing group. If you define these objects on a subsystem that is not the primary subsystem, you must also define that subsystem as an *additional subsystem* during customization.

All members of the data sharing group that run on the LPAR where DB2 UET is running must be included in the policy. You can use wildcards in the policy when specifying the SSID. For example, if members DB1A and DB1B are running on the same LPAR, in the policy, you can specify `DB1%` for the SSID.

So that the started task can communicate with the subsystems in a data sharing group, set the **DB2_CONNECT_TO_ALL_SUBSYSTEMS** initialization option for the started

task to YES. If you specify No, the started task can connect only to the subsystem that is specified in the DB2_SSID initialization option (that is, the primary subsystem).

Additional requirements, considerations, and restrictions depend on the DB2 UET interface that you use.

If you plan to use the DSNUTILB intercept, ensure that these requirements and recommendations are met:

- A DB2 UET started task must be running on each z/OS image where a member subsystem of the data sharing group is located. If some member subsystems are on a remote z/OS image, a started task configuration must be running on the remote z/OS image as well as on the z/OS image where the primary subsystem is located. Each started task must have its own set of DB2 objects and plans for DB2 UET. When you run the Tools Customizer for an additional started task configuration on a remote z/OS image, ensure that the field **The DB2 UET primary subsystem** on the **Product Parameters** panel contains the name of the DB2 SSID to use as the primary subsystem on the remote z/OS image.
- Each member subsystem must be specified in the DSNUTILB intercept policy for the local started task by using the <DB2SYSTEM> element.
- If the member subsystems are on different z/OS images, it is recommended that you associate the same intercept policy with the started task configuration on each of these z/OS images so that the same intercept policy will be used for all members of the data sharing group.

If you plan to use the ISPF interface or batch interface, only one started task configuration and one intercept policy are required, even if the member subsystems are on different z/OS images. All members of the data sharing group will share a common set of DB2 objects for DB2 UET: the audit and logging tables on the primary subsystem, and the thread-blocker table, DSNUTILB-intercept worklist tables, and mapping-table database and table space on any active subsystem in the data sharing group. You will be able to cancel threads on any active subsystem that is within the same data sharing group as the primary subsystem. However, the following restrictions apply:

- You cannot perform escalated cancelations of DB2 threads on any member subsystems that are located on a z/OS image other than the z/OS image where the primary subsystem is located.
- For batch thread-cancelation operations that include thread blocking, you cannot perform the job step for blocking and canceling threads on a member subsystem that is on one z/OS image and then perform the job step for allowing new threads on another member subsystem that is on a different z/OS image. That is, the BLOCK_THREADS and ALLOW_THREADS actions must apply to the same member subsystem on one z/OS image.

The following figure shows a sample product configuration in a data sharing environment:

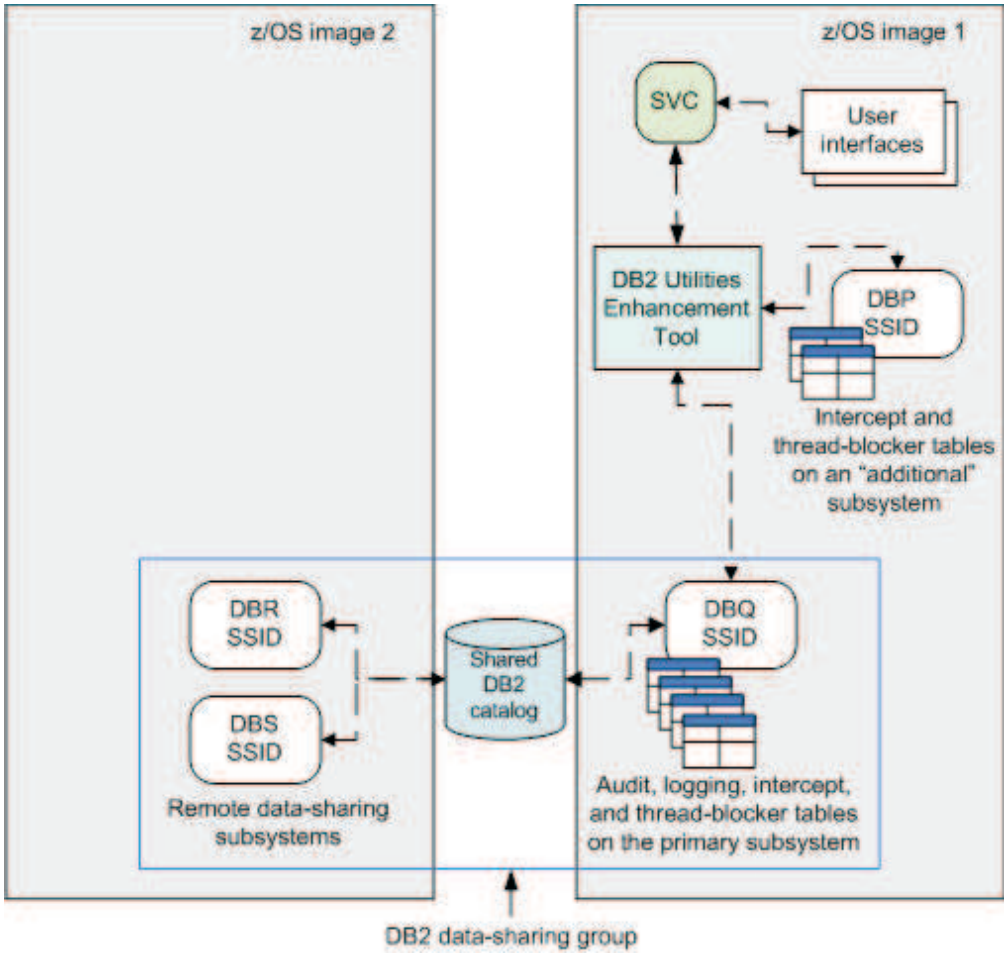


Figure 2. Sample product configuration in a DB2 data sharing environment

The DB2 subsystems DBQ, DBR, and DBS comprise a data sharing group that spans two z/OS images. DBQ is the primary DB2 subsystem for the DB2 UET started task configuration, and it contains

- audit and logging tables that are used for all started task operations, including those that involve the non-data sharing subsystem DBP
- thread-blocker and DSNUTILB-intercept tables, which are shared by all subsystems in the data sharing group

If you use the ISPF or batch interface, you could cancel threads on the DBP subsystem and on each subsystem in the data sharing group (DBQ, DBR, and DBS). However, if you use the DSNUTILB intercept, you could perform intercept processing (thread cancelation, or the utility-specific enhancements) only on the DBP and DBQ subsystems. To perform intercept processing on the DBR and DBS subsystems, another started task configuration would need to be running on z/OS image 2, and the intercept policy for that started task configuration would need to specify the DBR and DBS subsystems.

When you define the policy to run in a data sharing environment, you must list each DB2 subsystem; you cannot specify the group attach name.

Related concepts:

"DB2 environment" on page 16
DB2 UET runs as a started task on a z/OS system. The started task communicates with DB2 to perform product functions and to store information about product activities in DB2 tables.

Security requirements

Review the security requirements for DB2 Utilities Enhancement Tool.

If your site uses ACF2 to restrict TSO command use, you may need to add the TSO command that DB2 UET uses, ABPFMAIN, to the ACF2 Command Limiting table.

Authorization requirements for the started task

Make sure that the DB2 UET started task runs under a user ID that has the required authority. To control security at the user level, you must create a security exit.

The started task must run under a user ID that has SYSADM or SYSCTRL privileges.

RACF authority must be granted to the started task user ID to

- execute DFSMSdss to copy production data set data to the shadow object's data sets
- rename image copy data sets to the production objects' names

To control access to thread-management functions or ISPF panels at the user level, you can create a security exit in assembler language. If you do not create a security exit, all users can access all product functions. For example, you could create a security exit to control which users can perform thread cancelations and access the product administration panels. A sample security exit (ABPXSE00) and corresponding DSECT (ABPAPISE) are available in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specify during customization. You must specify the name of the security exit in the started task initialization options member by using the option SECURITY_EXIT.

Related reference:

"Security exit" on page 451
You can create an optional security user exit to control user access to DB2 UET ISPF panels and thread-management functions. The exit will apply to a single DB2 UET configuration. If you do not create a security exit, all users will have access to all panels and thread-management functions.

Authorization requirements for LOAD utility enhancements

DB2 UET creates shadow objects for use with the IFDISCARDS and SHRLEVEL REFERENCE options. DDL associated with the shadow objects is executed in the DB2 UET started task, and the load processing runs in its own address space.

To use these features, the user ID that submits the load job must have one of the following authorizations:

- SYSCTRL or SYSADM authority
- ownership of the table
- LOAD privilege for the database
- STATS privilege for the database (if the STATISTICS keyword is specified)

- DATAACCESS authority

In addition, the user ID that submits the load job must have authority to complete the following actions:

- run the load against the production table
- rename, copy, delete, and define the DB2 object page sets
- execute the DFSMSdss COPY command with the TOLERATE(ENQFAILURE) keyword to create the shadow data sets. As stated in the DFSMSdss Storage Administration documentation, when the RACF® FACILITY class is active and profile STGADMIN.ADR.COPY.TOLERATE.ENQF is defined, you must have READ access authority to use the COPY command.

Considerations for running multiple started tasks

A single started task is usually sufficient to handle multiple user requests from any of the DB2 UET interfaces to perform work on one or more DB2 subsystems. However, if you have a very high volume of activity, you can run multiple started tasks concurrently to handle the workload more efficiently.

If you run multiple concurrent started tasks, the SABPSAMP library must contain a separate started task initialization options member for each started task. Each initialization options member must specify a unique SVC number.

Also, each started task should have its own set of DB2 UET audit, logging, DSNUTILB-intercept, and thread-blocker tables. Although it is possible to share these tables across started tasks (because each row in each table is qualified by the *abpid* value for the started task configuration), this practice is not recommended. When multiple started tasks use the same tables, a very large volume of data might be written to the tables.

You will need to run the Tools Customizer for each started task to generate the following items: the DDL for creating the DB2 objects that the started task will use, the statements for binding the DB2 plan and packages on the DB2 subsystems that the started task will communicate with, the sample started task PROC, the started task initialization options member, and the sample DSNUTILB intercept policy. A separate DSNUTILB intercept policy is not required for each started task; the same intercept policy can be used by multiple started tasks if appropriate.

Tip: If you want to prevent a started task from performing work on a particular DB2 subsystem, you can *not* bind the plan and packages for the started task on that subsystem.

After you set up the started tasks, you can select the one that you want to use for performing product functions, as follows:

- In the ISPF interface, select a started task configuration ID on the Set ABPID panel.
- In a batch thread-cancelation job, specify a started task configuration ID by using the ABPID parameter.
- If you are using the DSNUTILB intercept, ensure that the ABPPLCY DD statement in the started task PROC specifies the DSNUTILB intercept policy that you want to use for performing intercept processing with that started task.

The following additional considerations apply if you are using the DSNUTILB intercept:

- A single started task can connect to multiple DB2 subsystems to perform intercept processing. You specify the subsystems in the DSNUTILB intercept policy for a started task by using the <DB2SYSTEM> element. However, any given DB2 subsystem can be intercepted by only one started task at a time.
- You can run multiple started tasks to perform intercept processing on different DB2 subsystems. You must specify a DSNUTILB intercept policy for each started task. If more than one started task policy identifies the same DB2 subsystem (either by specifying the same SSID or a wildcard pattern), the started task that connects to that subsystem first will have exclusive use of the subsystem until intercept processing completes. No other DB2 UET started task will be able access the subsystem while the first started task is performing intercept processing.

The following figure shows two DB2 UET started tasks performing DSNUTILB intercept processing for three subsystems:

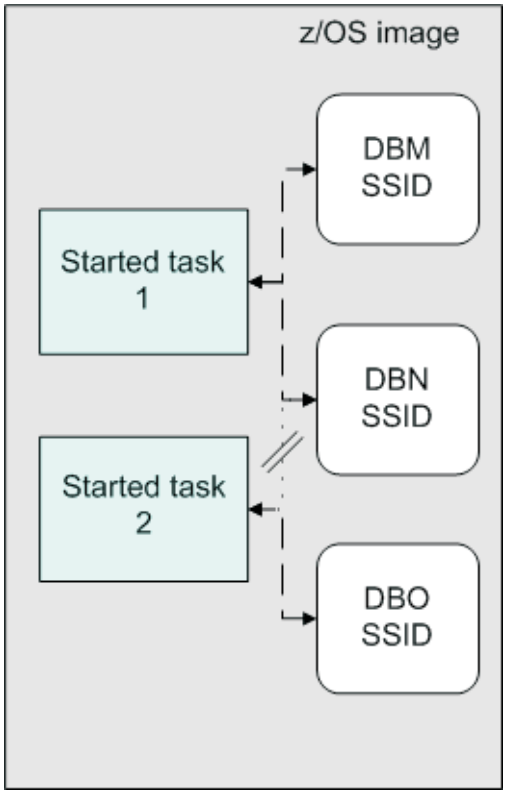


Figure 3. Using multiple started tasks for DSNUTILB intercept processing

The DSNUTILB intercept policy for started task 1 specifies the DB2 subsystems DBM and DBN. The policy for started task 2 specifies the DB2 subsystems DBN and DBO. Because started task 1 initializes first, it intercepts DBN before started task 2. Started task 2 cannot intercept DBN until after started task 1 completes its intercept processing on DBN.

Dispatching priority

Ensure that the dispatching priority for the DB2 UET task is set correctly with respect to other dispatching priorities. The dispatching priority determines the order in which a task can use the processor in a multitasking environment.

The DB2 UET dispatching priority must be lower than the priority values for the DB2 subsystems that DB2 UET will use but higher than the priority values for any utilities or batch jobs for which DB2 UET will perform processing. Set the dispatching priorities for these items in the following order (from highest to lowest priority):

1. The address spaces of the DB2 subsystems that DB2 UET will use (highest dispatching priority)
2. The DB2 UET started task
3. The DB2 utilities, batch jobs, and applications for which DB2 UET performs a function such as thread cancelation (any dispatching priority under the DB2 UET started task)

Use of WTO messages for automated operations

Consider whether you want to use the Write-to-Operator (WTO) messages that DB2 UET issues for automated operations.

DB2 UET issues some messages as WTO messages in addition to printing them to the SYSPRINT data set. If your automation tools can process WTO messages, you can use these messages to control the flow of automated operations in your environment.

The following WTO messages on the status of the DB2 UET started task are particularly useful: ABPS0001I, ABPS0002I, ABPS0003I, and ABPS0004I. These messages report the beginning and end of the started task initialization and termination phases.

Operational issue related to the vector table

Consider this operational issue regarding the common vector table.

Some IBM products for z/OS systems, including DB2 UET, share a 4-KB vector table that resides in ECSA (subpool 228, storage key 0). This table, called the *RVT*, serves as an anchor point for these products. The first product to start after an IPL will obtain and initialize the RVT. If that product subsequently terminates, the RVT storage is deliberately *not* released because it could be serving multiple products. A system-monitoring product might report the RVT storage as "orphaned" or "owner gone." In this situation, do *not* attempt to release the RVT storage. If you do so, the other products that are using the vector table can be severely damaged.

DB2 version migration considerations

This topic details what you need to do when a DB2 subsystem being used with DB2 UET is migrated to a later version, or when a fallback to an earlier version is required.

Migration instructions:

1. Start Tools Customizer.
2. Update the DB2 level parameter and the Mode parameter on the DB2 Parameters Panel with the correct version and mode of DB2.
3. Regenerate the JCL to create the correct BIND job.
4. Bind the packages and plan for the higher version of DB2 to which you migrated.

5. Update the DB2 library concatenation within the started task to contain the correct DB2 libraries.

Fallback instructions:

1. Start Tools Customizer.
2. Update the DB2 level parameter and the Mode parameter on the DB2 Parameters Panel with the correct version and mode of DB2.
3. Regenerate the JCL to create the correct BIND job.
4. Bind the packages and plan for the lower version of DB2 to which you are falling back.
5. Update the DB2 library concatenation within the started task to contain the correct DB2 libraries.

Related concepts:

Chapter 3, "Customizing the product," on page 71
After you install DB2 Utilities Enhancement Tool by following the installation instructions in the Program Directory, you must run Tools Customizer to specify the variables for each DB2 subsystem and to customize the configuration and user parameters.

Worksheets: Gathering required data set names

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

Data set names for Tools Customizer

Identify and record the following Tools Customizer data set names:

Data set name	Special requirements	Your data set name
SCCQDENU Metadata library for Tools Customizer	None	
SCCQLOAD Executable load module library for Tools Customizer	None	
SCCQMENU ISPF messages for Tools Customizer	None	
SCCQPENU ISPF panels for Tools Customizer	None	
SCCQSAMP Sample members for Tools Customizer	None	
SCCQTENU Table library for Tools Customizer	You must have write access to this data set.	

Data set names for DB2 Utilities Enhancement Tool

Identify and record the following DB2 Utilities Enhancement Tool data set names. During the customization process, you will enter the following values on panel CCQPPRD.

Data set name and description	Special requirements	Your data set name
SABPDBRM DBRM library for DB2 Utilities Enhancement Tool	None	
SABPLOAD Executable load module library for DB2 Utilities Enhancement Tool	You must APF authorize this data set.	
SABPMENU ISPF messages for DB2 Utilities Enhancement Tool	None	
SABPPENU ISPF panels for DB2 Utilities Enhancement Tool	None	
SABPSAMP Sample members for DB2 Utilities Enhancement Tool	None	
SABPDENU Metadata library for DB2 Utilities Enhancement Tool product parameters	None	

Data set names of other libraries used by Tools Customizer

Identify and record the following data set names. During the customization process, you will enter the following values on the Tools Customizer Settings panel (CCQPSET).

Data set name and description	Special requirements	Your data set name
Product customization library Contains the customization jobs that Tools Customizer generates for DB2 Utilities Enhancement Tool. To customize DB2 Utilities Enhancement Tool, submit the members of the data set in the order in which they are displayed on the Finish Product Customization panel. The data set naming convention is: *hlq.$LPAR-name$.xyzvrm* where: • *hlq* is the value of the Customization library qualifier field on the Tools Customizer Settings panel (CCQPSET) • *LPAR-name* is the four-character LPAR name • *xyzvrm* is the three-letter product identifier with the version, release, and modification level For example, the data set name might be DB2TOOL.PRODUCT.CUST.$MVS1$.XYZ410.	You must have write access to this data set.	
BBY load library data set Specifies the fully qualified library name for the DB2 Utilities Solution Pack load module (BBY$NMIC).	If the DB2 Utilities Solution Pack (BBY) was not purchased, this value should be left blank.	

Data set name and description	Special requirements	Your data set name
Discover output data set Contains the output that is generated when you run the DB2 Utilities Enhancement Tool Discover EXEC. The DB2 Utilities Enhancement Tool Discover EXEC retrieves the metadata and values for the parameters from a previous customization of DB2 Utilities Enhancement Tool. The default name of the data set is DB2TOOL.CCQ110. DISCOVER. You can change the default value on the Tools Customizer Settings panel or the Discover Customized Product Information panel.	You must have write access to this data set.	
Data store data set Contains product, LPAR, and DB2 parameter values, and DB2 entry associations. Tools Customizer uses this data set to permanently store all information that is acquired about the product, DB2 subsystems, and LPAR when you customize products on the local LPAR. The default name of the data set is DB2TOOL.CCQ110. DATASTOR. You can change the default value on the Tools Customizer Settings panel.	You must have write access to this data set.	

Worksheets: Gathering parameter values for Tools Customizer

During the customization process, you will need to provide parameter values for DB2 Utilities Enhancement Tool, for DB2, and for your LPAR.

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

Metadata library for DB2 Utilities Enhancement Tool

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

Parameter	Discovered?	Your value
Metadata library The default name of the metadata library after the product has been SMP/E installed is *hlq*.SABPDENU, where *hlq* is the high-level qualifier for DB2 Utilities Enhancement Tool.	No	

Customization values for the Discover EXEC

Use the following worksheet to identify and record the required customization values for the Tools Customizer Discover EXEC. The values in this worksheet are for extracting information from a product that has already been customized. During the customization process, you will enter these values on panel CCQPDSC.

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

Parameter	Default value	Your value
Discover EXEC library The fully qualified data set name that contains the product Discover EXEC.	*hlq.mlq*.SABPDENU	
Discover EXEC name The name of the Discover EXEC.	ABPDISCO	
Discover output data set The fully qualified name of the data set for the output from the product Discover EXEC.	The name that you specified in option 0 User Settings from the Tools Customizer main menu.	
DB2 UET V2.2 Configuration ID The configuration ID for which the current DISCOVER process is to be performed.	ABP1	
DB2 UET V2.1 SABPSAMP data set The SABPSAMP library from the previous installation of DB2 UET that was created as part of the SMP/E installation process.	ABP.V210.SABPSAMP	

Parameter	Default value	Your value
DB2 UET V2.1 options module name The customized options module name in the sample library of the previous installation of DB2 UET that is specified in the previous field.	No default	
Copy existing policy member? Specify Y to copy the existing policy member from DB2 UET v2.1 for use with DB2 UET V2.2. Specify N to create a default policy in the DB2 UET v2.2 SABPSAMP data set.	Y	
DB2 UET V2.1 policy name If you would like to copy the policy member already in use, provide the name of the policy that you would like to re-use from the previous installation of DB2 UET utility.	No default	
DB2 UET V2.2 data set HLQ The high-level qualifier for the DB2 UET V2.2 data sets, which you specified during installation, or when you copied the target libraries.	ABP.MLQ	

Product to Customize section

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

Parameter	Discovered?	Source of this value
Product metadata library The library that you specified on the Specify the Product to Customize panel. This field is scrollable. To view its full contents, place the cursor anywhere on the field and press PF11.	Yes	This value is specified on the Specify the Product to Customize panel (CCQPHLQ).
LPAR The **LPAR** field displays the LPAR on which you are customizing DB2 Utilities Enhancement Tool.	Yes	This value is supplied by Tools Customizer.

Parameter	Discovered?	Source of this value
Product name The name of the product that you are customizing, DB2 Utilities Enhancement Tool. should be displayed in this field. This field is scrollable. To view its full contents, place the cursor anywhere on the field and press PF11.	Yes	The default value DB2 UET is provided by the product metadata file.
Version The version, release, and maintenance level of the product that you are customizing in the format V*n*.R*n*.*nn*.	Yes	This value is provided by the product metadata file. The default value for this release is 2.2.0.
Product customization library The name of the data set in which the generated library customization jobs will be stored.	No	This value is derived from the user-specified customization library qualifier on the Tools Customizer Settings panel (CCQPSET).

Product Parameters: Common parameters

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

Note: Tools Customizer displays some parameters only after you have selected tasks or specified values on the Product Parameters panel. Therefore, you must first define a primary SSID on the DB2 Parameters panel, then select values on the Product Parameters panel. Return to the DB2 Parameters panel to review options that were added as a result of your specifications on the Product Parameters panel.

Parameter	Required?	Discovered?	Default value	Your value
DB2 UET plan name The name of the DB2 plan that DB2 UET uses. The plan name must be unique on the DB2 subsystem where the plan is bound or within the data sharing group to which that subsystem belongs.	Yes	No	No default	
DB2 UET plan qualifier The qualifier for the DB2 UET plan. This variable is also used as the collection ID for the DB2 UET packages and as the creator ID for the DB2 UET tables and indexes. **Restriction:** DB2 UET does not support double-byte character set (DBCS) characters in this value.	Yes	Yes	ABP22PLN	
DB2 UET data sets HLQ The DB2 UET high-level qualifier for the product data sets, which you specified at installation time or when you copied the target libraries.	Yes	Yes	No default	

Parameter	Required?	Discovered?	Default value	Your value
DB2 UET started task user ID The IBM RACF® ACF2 or TSS user ID under which the started task will run. This ID will also be used as the OWNER identifier when binding the plan and packages and as the CURRENT SQLID (SQL authorization ID) when creating the product's DB2 objects and when issuing GRANT statements during object creation and following bind processing. **Important:** Ensure that this user ID has SYSADM or SYSCTRL authority on each DB2 subsystem where the DB2 UET plan will be bound.	Yes	No	No default	
Solution Pack load library Specifies the fully qualified library name for the DB2 Utilities Solution Pack load module (BBY$NMIC). If the DB2 Utilities Solution Pack (BBY) with Autonomics Director was not purchased, leave this field blank.	No	No	No default	
DB2 UET configuration name An identifier for the DB2 UET started task configuration (also referred to as the ABPID). This value must be four alphanumeric characters long. You must specify an ABPID to access the ISPF interface and to create a batch job that blocks threads. This ID serves as the basis of the sample started task name that Tools Customizer generates and inserts into the started task PROC. (The started task name is used in various product commands.) Tools Customizer adds ABP to the beginning of the ABPID to create the sample started task name. For example, if the ABPID is ABP1, the sample started task name would be ABPABP1. You can tailor the sample started task name in the started task PROC, if necessary.	Yes	Yes	No default	

Parameter	Required?	Discovered?	Default value	Your value
Thread cancel member name Specifies the name of the member that contains the optional cancel-control parameters for any thread cancelations that you perform by using the DSNUTILB intercept. These options are: • CANCEL_TYPE • ESCALATE • CHECK_THDTERM _RETRY_COUNT • CHECK_THDTERM _RETRY_INTERVAL Tools Customizer generates the sample member *abpid*BCAN (where *abpid* is the started task configuration ID that you specified) in the *hlq.mlq*.SABPSAMP library for your use. If you change the name of this member, you must specify the new name in this option. The member must reside in the same partitioned data set (a PDS or PDSE) as the started task initialization options member (that is, the data set that is allocated by the ABPOPTS DD statement in the started task PROC).	Yes	No	ABP@BCAN	
Global parameters member name Specifies the name of the member that contains the optional global parameters for any thread cancelations that you perform by using the DSNUTILB intercept. These options are: • ESCAPE • EXEC_TYPE • ON_FAILURE, REPORT_TYPE • THREAD_QUIESCE_TIME When Tools Customizer ran, DB2 UET generated the sample member *abpid*BGLB (where *abpid* is the started task configuration ID that you specified) in the *hlq.mlq*.SABPSAMP library for your use. If you changed the name of this member, you must specify the new name in this option. This member must reside in the same PDS or PDSE as the started task initialization options member (that is, the data set that is allocated by the ABPOPTS DD statement in the started task PROC).	Yes	No	ABP@BGLB	

Parameter	Required?	Discovered?	Default value	Your value
The DB2 UET primary subsystem The subsystem identifier for the primary DB2 subsystem where the audit and logging tables reside. If you are customizing parameters for a Secondary Subsystem, then leave this value as the Primary Subsystem name.	Yes	Yes	No default	
SYSOUT class The SYSOUT class of the DB2 UET started task.	Yes	Yes	*	
Work database bufferpool The name of the buffer pool that you want to use for the mapping-table objects that are created for the extended functionality of the REORG TABLESPACE utility.	Yes	Yes	BP0	
Work database name The name of the database in which you want to store work objects that are created for the extended functionality of the CHECK DATA utility.	Yes	Yes	No default	
Work database storage group The name of the storage group that you want to use for the mapping-table objects that are created for the extended functionality of the REORG TABLESPACE utility.	Yes	Yes	SYSDEFLT	

Task: Create customized DB2 UET jobs

This required task must be selected before you can select any of the subtasks that are described in the following sections. To create JCL that customizes an installation of DB2 UET, you must have first defined product-specific parameters and have associated a DB2 SSID with this customization. The generated job names themselves might vary, however, the template names do not.

Parameter	Required?	Discovered?	Default value	Your value
SYSAFF for product customized jobs The SYSAFF parameter is generated into each customized batch job that must run on a specific LPAR.	No	No	No default	

Tasks: Create new/Update existing DB2 UET objects

Choose the appropriate task to create new or update existing required DB2 objects for use with DB2 UET for the subsystem on which DB2 UET was customized.

Template name
ABPOBJCR. You must run this template job once for each SSID on which you will use DB2 UET. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Parameter	Discovered?	Default value	Your value
DB2 UET system table spaces STOGROUP Specifies the name of the storage group that is used internally by DB2 UET for table spaces that contain audit and logging tables and the thread-blocker and DSNUTILB intercept tables. Valid values must follow standard DB2 naming conventions for storage groups.	No	SYSDEFLT	
DB2 UET system table spaces buffer pool Specifies the buffer pool that is used for table spaces that contain the DB2 UET audit and logging tables and the thread-blocker and DSNUTILB intercept tables. Valid values must follow standard DB2 naming conventions for buffer pools.	No	BP0	
DB2 UET system index spaces STOGROUP Specifies the name of the storage group that is used internally by DB2 UET for index spaces that contain indexes on audit and logging tables and the thread-blocker and DSNUTILB intercept tables. Valid values must follow standard DB2 naming conventions for storage groups.	No	SYSDEFLT	
DB2 UET system index spaces buffer pool Specifies the buffer pool that is used for index spaces that contain indexes on the DB2 UET audit and logging tables and the thread-blocker and DSNUTILB intercept tables. Valid values must follow standard DB2 naming conventions for buffer pools.	No	BP0	

Task: Create additional indexes

This optional (but highly recommended) task creates three additional indexes on DB2 catalog tables to improve the performance of DB2 catalog queries.

Template name
ABPIDXCR. If you decide to use the performance indexes, you must run this template job at least once for each SSID on which you will use DB2 UET. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Tip: Generate the job to see the columns that are needed for the additional performance indexes, but before running the job, check for a suitable existing index that can be used.
If any of the indexes that the DB2 UET packages depend upon are dropped, then those packages are invalidated, causing a REBIND the next time that the package is invoked.

Task: Create LOG/AUDIT Migration job

This optional task extracts data from your existing LOG and AUDIT tables and loads the data into the new tables. This task is recommended if you want to use fully populated tables on which to base the statistics and plan.

Template name
ABPMIGRT. This template, when submitted, will use different methods to extract data from your existing Log and Audit tables and load the data into the new tables. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Parameter	Required?	Discovered?	Default value	Your value
V2.1 LOG and AUDIT tables CREATOR name Enter the CREATOR name value for your existing tables.	Yes	No	ABP21	

Task: Create JOURNAL Migration job

This optional task extracts data from your existing JOURNAL tables and loads the data into the new tables. This task is recommended if you want to use fully populated tables on which to base the statistics and plan.

Template name
ABPMIGR2. This template, when submitted, will use different methods to extract data from your existing Journal tables and load the data into the new tables. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Step or parameter	Required?	Discovered?	Default value	Your value
V2.1 JOURNAL tables CREATOR name Enter the CREATOR name value for your existing tables.	Yes	No	ABP21	

Task: Create RUNSTATS job

This optional task runs the RUNSTATS utility to collect catalog statistics for the optional indexes on DB2 catalog tables. This task is recommended if you chose to create the three optional performance indexes on the DB2 catalog.

Template name
ABPRSTCR. You must run this template job at least once for each SSID on which DB2 UET performance indexes will be used. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Task: Create BIND job

This task binds the required DB2 UET packages and plans for the subsystem on which DB2 UET was customized.

Template name
ABPBNDCR. You must run this template job once for each SSID on which

DB2 UET will be used. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Task: Create cleanup job for a worklist table

This optional task creates SAMPLIB maintenance job *ssid*#ERR for the DB2 subsystem designated as the primary DB2 subsystem. *ssid* is the primary DB2 subsystem on which DB2 UET is being customized.

Job *ssid*#ERR is created for DB2 subsystems monitored by DB2 UET. The job contains SQL statements for manually deleting rows from the DSNUTILB-intercept worklist error tables that are older than the number of days specified. The job is stored in the *hlq.mlq*.SABPSAMP data set, where *hlq* and *mlq* are the product data set high-level qualifier and mid-level qualifier that were specified on the Product Parameters panel.

Template name
ABPWKLCR. You must run this template job at least once for each primary DB2 subsystem. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Task: Create the options module

This required task creates the options module *abpid*OPTS for use with the DB2 UET started task. *abpid* is the value that you specified in the **DB2 UET configuration name** field on the Product Parameters panel.

All parameters are required.

Template name
ABPOPTCR. This template is created once per DB2 UET configuration, and you must run it at least once for each DB2 UET configuration that you customize. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Parameter	Discovered?	Default value	Your value
STC audit active Controls whether DB2 UET records audit information for thread cancelation activities in a DB2 table. Valid values are YES and NO (default). The product administrator can temporarily override this setting by specifying another value in the **Audit status** field on the Control System panel.	Yes	YES	

Parameter	Discovered?	Default value	Your value
Maximum STC audit age Indicates the maximum number of days to retain rows for audit information in the audit table (ABPAUDIT). This number of days is counted from the time that the rows are inserted into the table. When a row reaches this age limit, it is eligible for deletion. It is automatically deleted from the table the next time that a new row is inserted into the table. Valid values are in the range 0 - 32767. The default value 0 prevents the automatic deletion of old rows from the audit table. If you accept the default value, you must manually delete old rows from the audit table periodically to prevent the table from becoming too large. Use the sample SQL that is provided in the SABPSAMP member *ssid*LOG, where *ssid* is the 4-character subsystem ID where the logging tables reside.	Yes	0	

Parameter	Discovered?	Default value	Your value
COPY command FASTREPLICATION keyword Specifies whether the use of DFSMSdss fast replication (FlashCopy®) is preferred, required, or not to be used. The value specified for this option does not affect concurrent copy or virtual concurrent copy processing. Valid values are as follows: • PREFERRED: (default) Specifies that you want to use a fast replication method, if possible. If fast replication cannot be used, DFSMSdss completes the operation using traditional data movement methods. • REQUIRED: Specifies that fast replication must be used. DFSMSdss stops processing the current data set if fast replication cannot be used. However, DFSMSdss continues processing the rest of the data sets using fast replication. When the DEBUG(FRMSG(MIN|SUM|DTL)) keyword is not specified, DFSMSdss still issues summarized information regarding why a fast replication method cannot be used as though DEBUG(FRMSG(SUMMARIZED)) had been specified. The DEBUG(FRMSG(MIN | SUM | DTL)) keyword determines the amount of information provided for why you cannot use a fast replication method. • NONE: Specifies that fast replication should not be used. DFSMSdss does not attempt to use fast replication and completes the operation using traditional data movement methods. For more information about FASTREPLICATION, see the explanation of COPY command keywords in the IBM *z/OS DFSMSdss Storage Administration* documentation.	Yes	PREFERRED	

Parameter	Discovered?	Default value	Your value
COPY command DEBUG keyword Specifies whether you want to use DEBUG as a diagnostic tool. Valid values are as follows: • NONE: DEBUG is not used. • FRMSG: DFSMSdss issues messages that explain why you cannot use fast replication or Preserve Mirror operation during COPY processing. Specify DEBUG(FRMSG) with one of the following sub-keywords: – FRMSG(MINIMAL): DFSMSdss issues a message with a minimal level of information. – FRMSG(SUMMARIZED): DFSMSdss issues an informational message with summary information. – FRMSG(DETAILED): DFSMSdss issues a message with detailed information When applicable, detailed information regarding ineligible volumes is provided in the message text. • SMSMSG: DFSMSdss displays ACS WRITE statements to the job output. For more information about DEBUG, see the explanation of COPY command keywords in the *IBM z/OS DFSMSdss Storage Administration* documentation.	Yes	NONE	
Escalated cancel active Controls whether DB2 UET users are allowed to perform escalated cancelations of threads. An escalated cancelation uses the z/OS cancel command to terminate the job, TSO user, or started task that is associated with the thread. Valid values are as follows: • YES: Allow escalated cancelations • NO: (default) Do not allow escalated cancelations (use only the DB2 -CANCEL THREAD command).	Yes	NO	

Parameter	Discovered?	Default value	Your value
Connect to all DB2 subsystems? Controls whether DB2 UET attempts to connect to all active DB2 subsystems on the z/OS system on which it is configured, or only to the DB2 subsystem that is specified in the DB2_SSID initialization option (the subsystem that contains audit and logging information). Valid values are as follows: • YES: (default) DB2 UET attempts to connect to all active DB2 subsystems by default. • NO: DB2 UET attempts to connect only to the primary subsystem that is specified in the DB2_SSID option. Only that subsystem is listed on the Set DB2 System panel in the ISPF interface. **Note:** If you specify NO and create a DSNUTILB intercept policy that specifies DB2 subsystems other than the primary subsystem, no intercept processing occur on those additional subsystems.	Yes	YES	
Connection idle timeout Specifies the maximum amount of time (in seconds) that the DB2 connection for a DB2 UET task can have no activity. When this time limit is reached, the connection to DB2 closes. Valid values are in the range 0 - 32767. The default value is 300. If you specify 0, this timeout option is disabled and will not cause an inactive connection to close. This timeout option does not apply to the subtask for the DB2 UET connection to the DB2 subsystem that is specified by the DB2_SSID option.	Yes	300	
DB2 tasks count Specifies the maximum number of z/OS tasks that DB2 UET can start for connection to a single DB2 subsystem. Valid values are in the range 1 - 2147483647. The default value is 2.	Yes	2	
DB2 task idle timeout Specifies the maximum amount of time (in seconds) that a subtask for a DB2 UET connection to DB2 can remain inactive after the connection closes (that is, after the DB2_CONNECTION_IDLE _TIMEOUT limit has been met). When this time limit is reached, the subtask ends. Valid values are from 0 through 32,767. If you specify 0, this timeout option is disabled and will not cause an inactive subtask to end. This timeout option does not apply to the subtask for the DB2 UET connection to the DB2 subsystem that is specified by the DB2_SSID option.	Yes	900	

Parameter	Discovered?	Default value	Your value
DSNUTILB retry count Specifies the number of times the started task will attempt utility execution, waiting for 100 milliseconds between attempts. If all retry attempts fail, utility execution fails. Valid values are in the range 0 - 2147483647. The value 0 is treated as 1.	Yes	100	
DSCOPY limit Specifies the maximum number of concurrent data set operations for LOAD prevalidate. To prevalidate DB2 UET, validates records in the SYSREC file against the check constraints and data types of the table that you specify in the LOAD utility syntax. If it detects errors in the load, you can fail the load or pause the load to examine the errors, and then decide whether to restart the load despite the errors. Valid values are in the range 0 - 250. If you specify the value 0, DB2 UET automatically determines the limit.	Yes	0	
STC logging active Controls whether DB2 UET logs messages about product performance and operations in its DB2 log table. Valid values are YES (default) and NO. The product administrator can temporarily override this setting by specifying another value in the **Logging status** field on the Control System panel.	Yes	YES	
Maximum STC log age Indicates the maximum number of days to retain rows for logged messages in the logging table (ABPLOG). This number of days is counted from the time that the rows are inserted into the table. When a row reaches this age limit, it is eligible for deletion. It is automatically deleted from the table the next time a new row is inserted into the table. Valid values are in the range 0 - 32767. The default value 0 prevents the automatic deletion of old rows from the logging table. If you accept the default value, you must manually delete old rows from the logging table periodically to prevent the table from becoming too large. Use the sample SQL that is provided in the SABPSAMP member *ssid*LOG, where *ssid* is the 4-character subsystem ID where the logging tables reside.	Yes	0	

Parameter	Discovered?	Default value	Your value
Override mapping table Indicates whether to replace existing MAPPINGTABLE specifications that you manually defined for the DB2 REORG TABLESPACE utility with the MAPPINGTABLE specifications that DB2 UET automatically generates for the utility. Valid values are YES and NO (default). Regardless of how you set this option, note that DB2 UET will always add a MAPPINGTABLE specification to a REORG TABLESPACE utility statement that includes SHRLEVEL CHANGE if that statement does not already contain an existing MAPPINGTABLE specification.	Yes	NO	
Post-cancel user exit Specifies the name of the user exit that you optionally created for performing processing that you determined is necessary after thread cancelations. For example, you might create a post-cancel exit to notify users when thread-cancelation processing completes. If you are not using a post-cancel exit, specify `NONE` (default).	Yes	NONE	
Pre-cancel user exit Specifies the name of the user exit that you optionally created for performing some processing that you determined is necessary prior to thread cancelations. For example, you might create a pre-cancel exit to determine the DB2 objects that an application or utility will need to access. If you are not using a pre-cancel exit, specify NONE (the default value).	Yes	NONE	
Security-cancel exit Specifies the name of the user exit that you optionally created for verifying user authority to perform thread-management functions such as blocking and canceling threads and to access specific ISPF panels. If you are not using a security exit, specify NONE (the default value).	Yes	NONE	
SVC number The DB2 UET supervisor call (SVC) number. This number must be an integer from 200 through 255. Check with your systems programmer to ensure that you choose an SVC number that is available. The SVC will be dynamically installed when the DB2 UET started task starts and will be dynamically removed when the started task stops.	Yes	255	

Parameter	Discovered?	Default value	Your value
Shadow database prefix Specifies a prefix for the names of shadow objects that are used with the IFDISCARDS and SHRELEVEL REFERENCE options in the LOAD utility. The prefix can be one to four characters, and must follow standard DB2 naming conventions for databases. For each invocation of the Load Prevalidate or LOAD REPLACE SHRLEVEL REFERENCE feature, the product generates a unique set of shadow object names by using the specified prefix and appending a numeric suffix.	Yes	ABPS	
Shadow schema Specifies the schema (creator) to be used for the shadow objects that the product creates for use with the IFDISCARDS and SHRELEVEL REFERENCE options in the LOAD utility. When executing the Load Prevalidate or the LOAD REPLACE SHRLEVEL REFERENCE feature, the product creates a unique set of shadow objects using the specified shadow database prefix in the object names. The product uses the value that you specify for shadow schema as the schema name for the shadow objects that it creates.	Yes	ABPSTC	
DB2 UET trace active Controls whether DB2 UET collects trace information. Specify YES (the default value) to enable tracing, or specify NO to disable tracing. A trace is a record of DB2 UET internal processing that is primarily used by Support for diagnosing a problem. The product administrator can temporarily override this setting by specifying the another value in the **Trace status** field on the Control System panel.	Yes	YES	
Size of trace table Specifies the size (in MB) of the table in which DB2 UET stores trace information. Valid values are 1 - 2147483647. A value of 0 results in no trace table allocation. A trace is a record of internal processing that is primarily used by Support for diagnosing a problem.	Yes	1	
Workfile data class The name of a valid SMS data class for the temporary DASD data sets that are allocated by DB2 UET, or the value NONE.	Yes	NONE	
Workfile management class The name of a valid SMS management class for the temporary DASD data sets that are allocated by DB2 UET, or the value NONE.	Yes	NONE	

Parameter	Discovered?	Default value	Your value
Workfile storage class The name of a valid SMS storage class for the temporary DASD data sets that are allocated by DB2 UET, or the value NONE.	Yes	NONE	
Workfile unit The unit name for the location where the temporary DASD data sets that are allocated by DB2 UET are stored. Specify a valid unit name of a storage device or the value NONE. You can specify VIO if VIO (virtual input/output) is supported on your system and you want the temporary data sets to reside entirely in paging storage to improve performance. Alternatively, you can specify SYSALLDA to use any available DASD device.	Yes	SYSALLDA	
Maximum worklist table age Specifies the maximum number of days to retain rows in the DSNUTILB intercept worklist-error tables. A DSNUTILB intercept worklist contains the enhanced SYSIN information for a DB2 utility and can be used for restart purposes if a utility terminates. Worklist data is moved to worklist-error tables for diagnostic use by Customer Support in certain situations. After rows in the worklist-error tables reach the specified age limit, they are eligible for deletion. The next time a new row is inserted into a worklist-error table, the rows that meet the age limit are deleted. You can specify a value from 0 through 32,767 for this option. The default value 0 prevents the deletion of old rows from the worklist-error tables based on this option. If you accept the default value, you might need to manually delete old rows from these tables periodically to prevent the tables from becoming too large. Use the sample SQL that is provided in the SABPSAMP member *ssid*#ERR, where *ssid* is the 4-character subsystem on which the tables reside.	Yes	0	
WTO ROUTCDE Specifies the routing code for write-to-operator (WTO) messages regarding DB2 UET operations. Routing codes identify the z/OS console to which to send WTO messages and are defined when DB2 is installed. Valid values are from 1 through 28. The product administrator can temporarily override this value by entering another routing code on the Control System panel.	Yes	11	

Task: Create SAMPLIB members

This task creates the following SAMPLIB members that are used by the DSNUTILB intercept to block and cancel threads. In each member name, *abpid* is the value that you specified in the **DB2 UET configuration name** field on the Product Parameters panel. Each member is created once per DB2 UET configuration that is being customized. Each member is stored in the *hlq*.*mlq*.SABPSAMP data set, where *hlq* and *mlq* are the product data set high-level qualifier and mid-level qualifier that were specified on the Product Parameters panel.

- *abpid*BCAN contains the optional thread-cancel parameters that the DSNUTILB intercept uses for each cancel request for a DB2 utility.
- *abpid*BGLB contains the optional global parameters that pertain to intercept processing activities that the DSNUTILB intercept performs for the started task configuration.
- *abpid*PROC contains the started task PROC for DB2 UET.
- *abpid*PLCY contains a sample policy for use with the started task for DB2 UET.

Template name

ABPMODCR. This template is created once per DB2 UET configuration, and you must run it at least once for each DB2 UET configuration that you customize. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Step or parameter	Required?	Discovered?	Default value	Your value
Reuse DB2 UET 2.1 policy If you specified the policy name on the Discover Customized Product Information panel, this option is set to Y. DB2 UET reuses the policy from the previous version. If you did not specify a policy name, this option is set to N. DB2 UET create a new policy.	Yes	No	N	

Step or parameter	Required?	Discovered?	Default value	Your value
Cancellation processing If you run the Discover EXEC, this value is populated from the existing thread cancel SAMPLIB member. Specify the type of cancelation processing for DB2 UET to use for threads that contain uncommitted units of work. Valid values are as follows. **Important:** Do not specify NOBACKOUT or NO_UR_CHECKING unless necessary. These cancel types are not supported for threads on a DB2 subsystem other than the local subsystem to which a connection is established based on the <DB2SYSTEM> element value in the intercept policy. In a DB2 data sharing environment, the options do not apply to other subsystems in the data sharing group. • BACKOUT: (default) Specify this value to have DB2 back out (roll back) any uncommitted work that exists in the current unit of recovery when a thread is canceled. Backout processing ensures that the data integrity of the updated DB2 objects is maintained. Any updates that occurred after the last commit operation are backed out. • NOBACKOUT: Use this value with caution; data integrity problems are possible in certain situations. This value causes DB2 UET to check for any outstanding units of recovery just before canceling threads but prevents DB2 from performing backout processing. If DB2 UET finds an outstanding unit of recovery for a thread, the thread is *not* canceled and return code RC12 is issued. Depending on the **ON_FAILURE** parameter in member *abpid*BGLB, the cancel request either terminates or continues to the next thread. By default, the request continues and cancels the other threads. If DB2 UET does not find an outstanding unit of recovery, the thread is canceled. In this situation, data integrity problems can occur if an outstanding unit of recovery is created for a thread after DB2 UET has determined that no outstanding units of recovery exist for the thread and before the thread is canceled. • NO_UR_CHECKING: Use this value with caution; data corruption in your database is likely in certain situations. This value causes threads to be canceled without backout processing or any checking for outstanding units of recovery before cancelation. All threads that match your thread-filtering criteria are canceled, even if outstanding units of recovery exist for them. This value is equivalent to using the DB2 NOBACKOUT option on the -CANCEL THREAD command. For more information about the DB2 NOBACKOUT option, see the *IBM DB2 for z/OS Command Reference*.	Yes	Yes	BACKOUT	

Step or parameter	Required?	Discovered?	Default value	Your value
Escalate thread cancel If you run the Discover EXEC, this value is populated from the existing thread cancel SAMPLIB member. Specify whether DB2 UET issues the z/OS CANCEL command to cancel a thread when the DB2 -CANCEL THREAD command fails to do so. Valid values are as follows: • YES: DB2 UET can issue the z/OS CANCEL command to terminate the TSO user, batch job, or started task from which the thread originated. This escalated cancelation processing is supported only for the following connection types: CAF, IMSDLIBT, RRSAF, and TSO. Also, in a DB2 data sharing environment, escalated cancelation is supported only for threads on a local subsystem that you specify in a DB2SYSTEM element of the intercept policy. • NO: (default) DB2 UET issues the DB2 -CANCEL THREAD command but does not escalate cancelation processing to the z/OS CANCEL command when the DB2 -CANCEL THREAD command fails to terminate a thread. Any threads that were not canceled remain active. **Important:** DB2 UET will not perform an escalated cancelation, even if you set the ESCALATE parameter to YES, if a thread has a connection type that is not supported for escalated cancelation processing or if the CANCEL_ESCALATION_ACTIVE option in the started task initialization options member is set to NO.	Yes	Yes	NO	
Check thread termination retry count If you run the Discover EXEC, this value is populated from the existing thread cancel SAMPLIB member. After issuing the DB2 -CANCEL THREAD command, DB2 UET checks the status of the canceled threads to determine whether DB2 has actually terminated them. If some threads are not yet terminated, DB2 UET continues to check the thread status until all of the canceled threads are actually terminated or until the maximum retry count that you set with this parameter is reached. If the maximum retry count is reached and a thread has still not been terminated, DB2 UET issues an error message with the appropriate return code. If you also specify `YES` for the option to escalate thread cancel, DB2 UET will then issue the z/OS CANCEL command to terminate the thread. Valid values are in the range 1 - 32767. **Tip:** To define how frequently thread-status checking occurs, set the customization option `Check thread termination retry interval`.	Yes	Yes	20	

Step or parameter	Required?	Discovered?	Default value	Your value
Check thread termination interval If you run the Discover EXEC, this value is populated from the existing thread cancel SAMPLIB member. Specify how often (in seconds) DB2 UET checks the status of the threads that have been canceled to determine whether they are actually terminated in DB2. Valid values are in the range 1 - 32767. **Tip:** To define the maximum number of times that DB2 UET attempts thread-status checking, set the customization option `Check thread termination retry count`.	Yes	Yes	3	
Escape character If you run the Discover EXEC, this value is populated from the existing thread cancel SAMPLIB member. Specify the escape character that you want to use to delimit a wildcard character in an intercept policy rule for selecting threads for cancelation when that character is used as an actual part of the data value and not as a wildcard. If you do not specify the escape character immediately before such a character, DB2 treats the character as a wildcard when performing pattern matching during thread filtering. The default escape character is + (plus sign). This parameter is ignored in environments that use a double-byte character set (DBCS). For more information about escape expressions and pattern matching, see the *IBM DB2 for z/OS SQL Reference*.	Yes	Yes	+	

Step or parameter	Required?	Discovered?	Default value	Your value
Execution mode If you run the Discover EXEC, this value is populated from the existing global options SAMPLIB member. Specify the execution mode for thread-cancelation processing. Valid values are as follows: • CHECKPARMS: DB2 UET validates the syntax of all input parameters that are specified without actually blocking and canceling any threads. The DB2 utility will still run although no threads are canceled. • EXECUTE: (default) Perform thread blocking and cancelation. All threads that are selected based on the EXCLUDE and INCLUDE rules that you define in the DSNUTILB intercept policy are blocked and canceled. • SIMULATE: Perform a trial run of thread-cancelation processing for a utility. DB2 UET simulates the cancelation of threads that are selected based on the EXCLUDE and INCLUDE rules that you define in the DSNUTILB intercept policy. The processing flow, messaging, and reporting are the same as with EXECUTE mode, but no threads are actually blocked and canceled. Use this option to pre-check which threads will be canceled and to troubleshoot potential errors. The DB2 utility will still run although no threads are canceled.	Yes	Yes	EXECUTE	

Step or parameter	Required?	Discovered?	Default value	Your value
On thread cancel failure If you run the Discover EXEC, this value is populated from the existing global options SAMPLIB member. Specify whether thread-cancelation processing continues or terminates when an error occurs. Valid values are as follows: • CONTINUE: (default) Thread-cancelation processing continues when an error occurs. • TERMINATE: The job terminates and no additional cancelation processing occurs. For thread-cancel operations that include thread blocking, you can control whether DB2 UET reinstates the original statuses of the DB2 objects when cancelation processing fails or one or more of the canceled threads do not actually terminate. During thread blocking, DB2 UET changes the statuses of the DB2 objects to either RO (read only) or UT (only the utility for which threads are being canceled has access). To control whether DB2 UET leaves the object statuses in this state after a cancel failure, specify one of the following values: • TERMINATE RESET_OBJECT_STATUS YES: DB2 UET reinstates the original statuses of the DB2 objects when cancelation processing fails or one or more of the canceled threads do not actually terminate. • TERMINATE RESET_OBJECT_STATUS NO: DB2 UET retains the object statuses that were in effect during thread blocking.	Yes	Yes	CONTINUE	

Step or parameter	Required?	Discovered?	Default value	Your value
Report type If you run the Discover EXEC, this value is populated from the existing global options SAMPLIB member. Specify the level of detail in reports that DB2 UET provides for actual or simulated runs of thread-cancelation processing. Valid values are as follows: • SUMMARY: (default) Print the Threads Canceled report and the Threads Canceled Unit of Recovery report for each cancel request for which active threads are selected for cancelation. • DETAIL: Print the following reports for each cancel request: – Threads Canceled – Threads Canceled Unit of Recovery – All Active Threads – All Active Threads Unit of Recovery – All Active Threads Objects Referenced If no threads are selected for cancelation processing for a cancel request, DB2 UET generates only the reports on all active threads.	Yes	Yes	SUMMARY	
Thread quiesce time If you run the Discover EXEC, this value is populated from the existing global options SAMPLIB member. For thread-blocking actions, specify the number of seconds that DB2 UET waits between initiating thread blocking and canceling the active threads. This delay allows applications that are using the existing threads to complete units of work or quiesce before thread cancelation. The default value 0 causes DB2 UET to cancel existing threads immediately after initiating thread blocking.	Yes	Yes	0	

Task: Create maintenance members

This optional task creates the following SAMPLIB members. In each member name, *abpid* is the value that you specified in the **DB2 UET configuration name** field on the Product Parameters panel. Each member is created once per DB2 UET configuration that is being customized.

- *abpid*COMX, which contains a utility that resets DB2 UET COMX control blocks. Use this utility only when IBM Software Support directs you to do so.
- *abpid*MAIN, which contains the job that runs the DB2 UET batch interface program.
- *abpid*MNT, which contains the job that runs the ABPMAINT utility. This utility maintains the intercept worklist tables on the primary SSID. You can use the utility to

- terminate a DB2 utility for which DSNUTILB interception has occurred and to remove the worklist data that is associated with the utility ID
- restart a DB2 utility from the appropriate point after it ended because of exceptional circumstances that prevented a normal DB2 restart

Task **Create maintenance members** also creates the following SAMPLIB members for DB2 subsystems that are designated as the primary subsystem:

- *ssid*#LOG, which contains SQL statements for manually deleting rows from the log tables that are older than the number of days specified.
- *ssid*#AUD, which contains SQL statements for manually deleting rows from the audit table that are older than the number of days specified.

All members are stored in the *hlq.mlq*.SABPSAMP data set, where *hlq* and *mlq* are the product data set high-level qualifier and mid-level qualifier that were specified on the Product Parameters panel.

Template name
ABPSMPCR. This template is created once per DB2 UET configuration, and must be run at least once for each DB2 UET configuration that is being customized. The generated job is stored in the Product Customization Library that is displayed on the Finish Product Customization panel.

Task: Create product CLISTs

This optional task creates CLIST members ABPRUN and ABPF.

Template name
ABPRUNCR. This template is created once per DB2 UET configuration and is required to be run at least once. This generated job is stored in the **Product Customization Library** that is displayed on the Finish Product Customization panel.

DB2 Parameters

This section contains required DB2 parameters. During the customization process, enter these values on the DB2 Parameters panel (CCQPDB2).

You can create a DB2 entry as the primary subsystem or secondary subsystem and associate it with DB2 UET. When customizing DB2 UET, you must first define a primary subsystem before you can define product parameters.

You can customize DB2 UET only on DB2 entries that are associated with DB2 UET. The list of DB2 entries is on the Customizer Workplace panel. You can customize any associated DB2 entries for DB2 UET.

Note: Tools Customizer displays some parameters only after you have selected tasks or specified values on the Product Parameters panel. Therefore, you must first define a primary SSID on the DB2 Parameters panel, then select values on the Product Parameters panel. Return to the DB2 Parameters panel to review options that were added as a result of your specifications on the Product Parameters panel.

Parameter	Discovered?	Default value	Your value
DB2 subsystem ID The name of the DB2 subsystem you want to associate with DB2 UET. The value must be 4 characters or less. An example of a DB2 subsystem name is DB01.	No	No default	
Group attach name The name that is used by the TSO/batch attachment, the call attachment facility (CAF), DL/I batch, utilities, and the Resource Recovery Services attachment facility (RRSAF) as a generic attachment name. An example of a group attach name is DSG1.	No	No default	
Is this DB2 subsystem the primary subsystem? Specify whether this DB2 SSID is to be used as the primary subsystem or a secondary subsystem. If this is the primary subsystem, auditing and logging tables are created for DB2 UET on this subsystem. If this DB2 SSID is being configured as a secondary subsystem, auditing and logging tables are not created on this subsystem.	No	NO	
Use DBCS with this SSID? Specify whether this DB2 SSID is defined for use with double-byte character sets.	No	NO	
DB2 UET database name The name of the database (up to eight alphanumeric characters) that will be used for the DB2 UET auditing and logging tables.	No	No default	
Drop DB2 UET database first? Specify whether to create a DROP DATABASE statement within the SQL to first drop any previously created database. This option is helpful if you have already customized DB2 UET V2.2 and want to drop the database that was previously created.	No	YES	
Free product packages and plans Specify whether to add the FREE step to the BIND job to free DB2 UET version 2.2 packages and plans before running the product BIND.	No	YES	

Parameter	Discovered?	Default value	Your value
Drop DB2 UET work database first? Specify whether to create a DROP DATABASE statement within the SQL to first drop any previously created databases are used for work objects that DB2 UET creates. This option is helpful if you are re-customizing DB2 UET and want to drop the work database that was created before.	No	YES	
Mode The mode in which the DB2 subsystem is running. Valid values are CM, CM8, CM9, and NFM. CM is compatibility mode on V8. CM8 is conversion mode on V9 and V10. CM9 is conversion mode on V10. NFM is new-function mode on any DB2 version.)	No	No default	
Level number The version, release, and modification level of the DB2 subsystem. Valid values are 810, 910, and 101.	No	No default	
Load library The fully qualified data set name of the DB2 load library.	No	No default	
Run library The fully qualified data set name of the DB2 run library.	No	No default	
Exit library The fully qualified data set name of the DB2 exit library.	No	No default	
Plan name for the DSNTEP2 utility The name of the plan (up to eight alphanumeric characters) that is used for the DB2 DSNTEP2 program.	No	No default	

Starting and preparing Tools Customizer for use

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

Starting Tools Customizer

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

Before you begin

Tools Customizer must be SMP/E installed. You must know the high-level qualifier of where the Tools Customizer libraries reside. The high-level qualifier is

considered to be all the segments of the data set name except the lowest-level qualifier, which is SCCQEXEC.

About this task

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

Procedure

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

   You have to change the placeholder for the high-level qualifier only once. When you run the REXX EXEC, you do not have to pass the high-level qualifier as a parameter.
2. Run the REXX EXEC (CCQTCZ):
   a. From the ISPF Primary Option Menu, select option `6`. The ISPF Command Shell panel is displayed.
   b. Specify the EX command to run the REXX EXEC. For example, if the Tools Customizer EXEC library is CCQTCZ.USABSAND.SCCQEXEC and you changed the placeholder for the high-level qualifier in the REXX EXEC, specify: `EX 'CCQTCZ.USABSAND.SCCQEXEC(CCQTCZ)'`

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

Results

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

What to do next

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

Modifying Tools Customizer user settings

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

Procedure

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

```
CCQPSET            Tools Customizer Settings               14:03:51
Command ===>
Enter the settings for customizing a product or press End to save and exit.

Commands: SAVE - Save user settings

Product Customization Settings
  Customization library qualifier . . DB2TOOL.PRODUCT.CUST
  Use DB2 group attach  . . . . . . . YES (YES/NO)

Tools Customizer Library Settings
  Metadata library . . . . . DB2TOOL.CCQ110.SCCQDENU
  Discover output data set . DB2TOOL.CCQ110.DISCOVER
  Data store data set  . . . DB2TOOL.CCQ110.DATASTOR

User Job Card Settings for Customization Jobs
  ===> //         JOB
  ===>
  ===>
  ===>
  ===>
```

Figure 4. The Tools Customizer Settings panel (CCQPSET)

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

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

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

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

   Use DB2 group attach
   Determines the value that is used in the CONNECT statements in the generated customization jobs. Specify `YES` for data sharing environments, which causes the group attach name to be used. Specifying `NO`, in most cases, causes the SSID to be used in the DB2 CONNECT statement.

   Important: This field has no effect when you are customizing a product on a DB2 subsystem that is not a member of a data sharing group. In this case, the DB2 subsystem ID (SSID) is always used in the CONNECT statements in the generated customization jobs.

When you are customizing a product on a DB2 subsystem that is a member of a data sharing group, how the DB2 subsystem is defined and the value of the **Use DB2 group attach** field determines the value that is used in the CONNECT statements in the generated jobs. The following table shows whether the SSID or the group attach name is used:

Table 2. The effect of the value of the **Use DB2 group attach** *field in a data sharing environment*

DB2 subsystem definition	Value of the Use DB2 group attach field	Value that is used in the CONNECT statements
The DB2 subsystem is defined with an SSID.	Yes	Group attach name
	No	SSID[1]
The DB2 subsystem is not defined with an SSID.	Yes or No	Group attach name

Note 1: If you generate jobs for multiple DB2 subsystems that are defined with an SSID and belong to the same data sharing group, the SSID of the first DB2 subsystem that is selected is used.

For example, assume that on the Customizer Workplace panel, you generated jobs for the following DB2 subsystems:

- V91C, which is a stand-alone DB2 subsystem
- V91A, which is a DB2 subsystem that is a member of data sharing group DSG1
- A DB2 subsystem that was not defined with an SSID that is a member of data sharing group DSGA

The following figure shows how these DB2 entries might be listed on the Customizer Workplace panel:

```
Associated DB2 Entries and Parameter Status
  Line commands: G - Generate jobs  E - Edit  B - Browse  C - Copy  R - Remove
  Cmd SSID GrpAttch Lvl Mode User ID Date       Status             Message
      V91C  --      910 NFM  SYSADM  2010/11/09 Ready to Customize
      V91A  DSG1    910 NFM  SYSADM  2010/11/09 Ready to Customize
      --    DSGA    910 NFM  SYSADM  2010/11/09 Ready to Customize
------------------------------ End of DB2 entries -----------------------------
```

The following table shows which values are used in the CONNECT statements in the generated jobs, based on the value of the **Use DB2 group attach** field.

Table 3. Value that is used in the CONNECT statements in the generated jobs

SSID	GrpAttch	Value of the Use DB2 group attach field	Value that is used in the CONNECT statements
V91C	--	Yes	SSID
		No	SSID
V91A	DSG1	Yes	Group attach name
		No	SSID
--	DSGA	Yes	Group attach name
		No	Group attach name

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

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

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

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

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

```
//          JOB
```

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

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

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

Results

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

What to do next

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

Chapter 3. Customizing the product

After you install DB2 Utilities Enhancement Tool by following the installation instructions in the Program Directory, you must run Tools Customizer to specify the variables for each DB2 subsystem and to customize the configuration and user parameters.

For a summary of the customization steps, see the checklist in Chapter 2, "Preparing to customize," on page 23.

Topics:

Customizing DB2 UET

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

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

Use the following table to determine which roadmap corresponds to your environment.

Table 4. Customization roadmaps

Environment description	Roadmap
You do not have a customized version of DB2 UET, and you need to customize it for the first time.	"Roadmap: Customizing DB2 UET for the first time" on page 72
You have already customized a version of DB2 UET, and you want to use the same parameter values to customize a different version.	"Roadmap: Customizing a new version of DB2 UET from a previous customization" on page 72
You have a customized version of of DB2 UET, but you want to change one or more parameter values.	"Roadmap: Recustomizing DB2 UET" on page 73

Roadmap: Customizing DB2 UET for the first time

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

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

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

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

Table 5. Steps for customizing DB2 UET for the first time

Step	Description	Instructions
1	Specify the metadata library for the product that you want to customize.	"Specifying the metadata library for the product to customize" on page 74
2	Create new DB2 entries and associate them with DB2 UET.	"Creating and associating DB2 entries" on page 78
3	Define the required parameters.	"Defining parameters" on page 80
4	Generate the customization jobs for the product or for the DB2 entries on which DB2 UET is ready to be customized.	"Generating customization jobs" on page 84
5	Submit the generated customization jobs.	"Submitting customization jobs" on page 85

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

Table 6. Administrative tasks

Description	Instructions
Browse the different types of parameters.	"Browsing parameters" on page 87
Copy an existing DB2 entry to the list of DB2 entries on which DB2 UET can be customized.	"Copying DB2 entries" on page 87
Remove one or more DB2 entries from the associated list.	"Removing DB2 entries" on page 89
Delete one or more DB2 entries from the master list.	"Deleting DB2 entries" on page 89
Display a list of customization jobs that have been previously generated.	"Displaying customization jobs" on page 89
Maintain the customization jobs in the customization library.	"Maintaining customization jobs" on page 90

Roadmap: Customizing a new version of DB2 UET from a previous customization

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

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

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

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

Table 7. Steps for customizing a new version of DB2 UET from a previous customization

Step	Description	Instructions
1	Specify the metadata library for the product that you want to customize.	"Specifying the metadata library for the product to customize" on page 74
2	Use the DB2 UET Discover EXEC to discover information about the version of DB2 UET that you previously customized manually.	"Discovering DB2 UET information automatically" on page 76
3	Define the required parameters.	"Defining parameters" on page 80
4	Generate the customization jobs for the product or for the DB2 entries on which DB2 UET is ready to be customized.	"Generating customization jobs" on page 84
5	Submit the generated customization jobs.	"Submitting customization jobs" on page 85

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

Table 8. Administrative tasks

Description	Instructions
Browse the different types of parameters.	"Browsing parameters" on page 87
Copy an existing DB2 entry to the list of DB2 entries on which DB2 UET can be customized.	"Copying DB2 entries" on page 87
Remove one or more DB2 entries from the associated list.	"Removing DB2 entries" on page 89
Delete one or more DB2 entries from the master list.	"Deleting DB2 entries" on page 89
Display a list of customization jobs that have been previously generated.	"Displaying customization jobs" on page 89
Maintain the customization jobs in the customization library.	"Maintaining customization jobs" on page 90

Roadmap: Recustomizing DB2 UET

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

The new customization jobs will replace the customization jobs that were previously generated and stored in the customization library. Part of the recustomization process includes selecting or deselecting optional tasks or steps,

changing the definitions of parameters that have already been defined, or both. Use the method in this roadmap instead of deleting customization jobs from the customization library.

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

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

Complete the steps in the following table to recustomize DB2 UET.

Table 9. Required steps for recustomizing DB2 UET

Step	Description	Instructions
1	Specify the metadata library for the product that you want to recustomize.	"Specifying the metadata library for the product to customize"
2	Edit the specific tasks, steps, or parameters that need to be changed.	• "Defining DB2 UET parameters" on page 80 • "Defining DB2 parameters" on page 82
3	Generate the customization jobs for the product or for the DB2 entries on which DB2 UET is ready to be customized.	"Generating customization jobs" on page 84
4	Submit the new generated customization jobs.	"Submitting customization jobs" on page 85

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

Table 10. Administrative tasks

Description	Instructions
Browse the different types of parameters.	"Browsing parameters" on page 87
Copy an existing DB2 entry to the list of DB2 entries on which DB2 UET can be customized.	"Copying DB2 entries" on page 87
Remove one or more DB2 entries from the associated list.	"Removing DB2 entries" on page 89
Delete one or more DB2 entries from the master list.	"Deleting DB2 entries" on page 89
Display a list of customization jobs that have been previously generated.	"Displaying customization jobs" on page 89
Maintain the customization jobs in the customization library.	"Maintaining customization jobs" on page 90

Specifying the metadata library for the product to customize

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

About this task

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

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

Procedure

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

```
CCQPHLQ                Specify the Metadata Library             13:09:50

 Type the name of the metadata library for the pack or the product in the
 Metadata library field, or select the library in the list of previous
 libraries and press Enter to populate the field. Press Enter to continue.

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

 Metadata library . ABP.ALIAS.SABPDENU

 Previously Used Metadata Library:

 =>
 =>
 =>
 =>
```

Figure 5. The Specify the Metadata Library panel

2. Use one of the following methods to specify the product metadata library:
   - Type the name of a fully qualified partitioned data set (PDS) or an extended partitioned data set (PDSE) in the **Metadata library** field. Double quotation marks (") cannot be used around the name. Single quotation marks (') can be used but are not required. If you are customizing DB2 UET for the first time, you must use this method.
   - Place the cursor on the library name in the Recent Metadata Libraries list, and press Enter.

Results

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

What to do next

- Complete the steps that correspond to your environment:

  Customizing DB2 UET for the first time
  Do not run the DB2 UET Discover EXEC. Press End. The Customizer Workplace panel is displayed. If your environment requires associated

DB2 entries, ensure that they are created and associated. If your environment does not require associated DB2 entries, skip this step, and edit DB2 UET parameters.

Customizing DB2 UET from a previous or current customization
Press Enter to run the DB2 UET Discover EXEC. The Discover Customized Product Information panel is displayed. Specify the required information for running the EXEC.

Discovering DB2 UET information automatically

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

About this task

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

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

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

Procedure

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

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

   The Discover Customized Product Information panel is displayed, as shown in the following figure:

```
CCQPDSC            Discover Customized Product Information          14:20:49
  Command ===>                                                 Scroll ===> PAGE

  For the product you are customizing, the Discover EXEC retrieves product
  information from an already customized product. Specify the required
  information. To save your information and run the Discover EXEC, issue the RUN
  command. To save your information and stay on this panel, issue the SAVE
  command. To verify the syntax of your information without saving it, press
  Enter. To save and exit, press  End.

  Commands: RUN  SAVE

  Product to Customize
    Product metadata library : ABP.ALIAS.SABPDENU        >  LPAR. . . : RS23
    Product name . . . . . . : DB2 Utilities Enhancement  >  Version . : 2.2.0

  Discover EXEC for Extracting Information from an Already Customized Product
   Discover EXEC library  . . . ABP.ALIAS.SABPDENU
   Discover EXEC name . . . . . ABPDISCO
   Discover output data set . . TWPAPPA.CCQ.DISCOVER

  Information for Discover EXEC
   *DB2 UET V2.2 Configuration ID  . . . . . . ABP1
   *DB2 UET V2.1 SABPSAMP data set
      RSQA.ABP220.PI56503.SABPSAMP
   *DB2 UET V2.1 options module name . . . . . ABPOPTS
   *Copy existing policy member? . . . . . . . Y (Y, N)
    DB2 UET V2.1 policy name . . . . . . . . . ABPPLCY
   *DB2 UET V2.2 data set HLQ
      RSQA.ABP220.PI56503
```

Figure 6. The Discover Customized Product Information panel

2. Either accept the default values for the following input fields that Tools Customizer generates, or replace the default values with your own values:

   Discover EXEC library
   The fully qualified data set name that contains the DB2 UET Discover EXEC.

   Discover EXEC name
   The name of the DB2 UET Discover EXEC.

   Discover output data set
   The fully qualified data set where output from the DB2 UET Discover EXEC is stored.
3. Either accept or change the default values in the **Information for Discover EXEC** fields. These fields are generated by DB2 UET. They show the information that is required to run the DB2 UET Discover EXEC.
4. Issue the `RUN` command to run the DB2 UET Discover EXEC. Alternatively, save your information without running the DB2 UET Discover EXEC by issuing the `SAVE` command. If you issue the `RUN` command to run the DB2 UET Discover EXEC, the parameter information is discovered for DB2 UET, and the Customizer Workplace panel is displayed.

Results

The discovered parameter values for DB2 UET replace any existing values.

What to do next

The next step depends on your environment:

- If DB2 entries were not discovered, or if you need to customize DB2 UET on new DB2 entries, create and associate the entries.

- If DB2 entries were discovered and you want to customize DB2 UET on only these entries, define the parameters.

Related tasks:

"Creating and associating DB2 entries"
You can create new DB2 entries and associate them with DB2 UET.

"Defining parameters" on page 80
To customize DB2 UET, you must define DB2 UET parameters and DB2 parameters, if your customization requires DB2 entries.

Creating and associating DB2 entries

You can create new DB2 entries and associate them with DB2 UET.

About this task

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

Procedure

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

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

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

 Commands:  CREATE - Create a new DB2 entry

 Product to Customize
   Product metadata library : ABP.ALIAS.SABPDENU > LPAR  . . : MVS1
   Product name . . . . . : DB2 Utilities Enhancement Tool
   Product version  . . . . : 2.2.0

 Line commands: A - Associate  C - Copy

 Cmd SSID GrpAttch
 ----------------------------- End of DB2 entries -----------------------------
```

Figure 7. The Associate DB2 Entry for Product panel

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

```
CCQPDCR         Create a DB2 Entry
 Command ===>

 Specify a DB2 subsystem ID, a DB2 group attach name, or both for the
 new DB2 entry. Press Enter to continue or End to cancel.

  New DB2 Entry Information
   DB2 subsystem ID . . . . .
   DB2 group attach name  . .
```

Figure 8. The Create a DB2 Entry panel

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

   Tips:

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

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

```
CCQPDAD                Associate DB2 Entry for Product        Row 1 to 1 of 1
Command ===>                                                 Scroll ===> CSR

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

Commands:  CREATE - Create a new DB2 entry

Product to Customize
  Product metadata library : ABP.ALIAS.SABPDENU > LPAR  . . : MVS1
  Product name . . . . . : DB2 Utilities Enhancement Tool
  Product version  . . . . : 2.2.0

Line commands: A - Associate  C - Copy

Cmd SSID GrpAttch
    DB02   --
------------------------------ End of DB2 entries -----------------------------
```

Figure 9. The Associate DB2 Entry for Product panel with a new DB2 entry in the master list

c. Repeat steps b and c for each DB2 entry that you want to create.
d. When you have created all the DB2 entries, associate them with DB2 UET, or press End to display the Customizer Workplace panel.

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

Results

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

What to do next

Define the parameters.

Related concepts:

"Tools Customizer terminology" on page 441
Tools Customizer uses several unique terms that you should be familiar with

before you begin to use Tools Customizer.

Defining parameters

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

About this task

You must define the DB2 UET parameters first for the following reasons:

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

Defining DB2 UET parameters

DB2 UET parameters are specific to DB2 UET.

About this task

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

Whether you ran the DB2 UET Discover EXEC or you are manually editing the DB2 UET parameters, you must specify values for the **DB2 UET Started Task User ID** field and the **DB2 UET Plan Qualifier** field.

Procedure

1. Specify `E` next to the **Product parameters** field on the Customizer Workplace panel, and press Enter. The Product Parameters panel is displayed, as shown in the following figure. If other steps must be completed in a specific sequence before you define the DB2 UET parameters, a note labeled **Important** will display the correct sequence on this panel.

```
CCQPPRD                        Product Parameters                    13:57:25
Command ===>                                                  Scroll ===> PAGE

Complete the following tasks to customize the products. The required tasks,
required steps within a required task or selected task, and required
parameters are preceded by an asterisk (*). Ensure that values are specified
for the required parameters. Press End to save and exit.

Commands: SAVE VERIFYOFF
Line Commands: / - Select

Product to Customize
  Product metadata library : RSQA.ABP220.DR110628.SABPD >  LPAR. . . : RS25
  Product name . . . . . . : DB2 Utilities Enhancement  >  Version . : 2.2.0

Product customization library : CSJENN.TESTING2.$RS25$.ABP220
                                                                  More:     +

Usage Notes:
  If you did not run the Discover EXEC to use values from a previous version
  of DB2 Utilities Enhancement Tool, you must define a primary DB2 subsystem
  before you edit the parameters on this panel.

 Common parameters
  *DB2 UET plan name  . . . . . . . . . . . . ABP22PQA
  *DB2 UET plan qualifier . . . . . . . . . . ABP22
  *DB2 UET data sets HLQ  . RSQA.ABP220.PI56503
  *DB2 UET started task user ID . . . . . . . ABPSTC
   Solution Pack load library
     RSQA.BBY210.IBMTAPE.SBBYLOAD
  *DB2 UET configuration name . . . . . . . . QA1B
  *Thread cancel member name  . . . . . . . . ABP@BCAN
  *Global parameters member name  . . . . . . ABP@BGLB
  *The DB2 UET primary subsystem  . . . . . . QA1B
  *SYSOUT class . . . . . . . . . . . . . . . *
  *Work database bufferpool . . . . . . . . . BP0
  *Work database name . . . . . . . . . . . . ABP22TDB
  *Work database storage group  . . . . . . . SYSDEFLT

/ Create customized DB2 UET jobs
   SYSAFF for product customized jobs . . . . RS25

   / Create new DB2 UET objects
    *DB2 UET system table spaces STOGROUP . . SYSDEFLT
    *DB2 UET system table spaces buffer pool  BP0
    *DB2 UET system index spaces STOGROUP . . SYSDEFLT
    *DB2 UET system index spaces buffer pool  BP0
    *DB2 UET system table spaces buffer pool  BP0
    *DB2 UET system index spaces STOGROUP . . SYSDEFLT
    *DB2 UET system index spaces buffer pool  BP0

   / Create additional indexes (optional)
```

Figure 10. The Product Parameters panel

2. Select any required tasks and steps, and specify values for any parameters. After you select a task or step with a slash (/), put the cursor in the selected field and press Enter. If tasks, steps, and parameters are required, they are preselected with a slash (/). Otherwise, they are not preselected. If you are customizing a new version of DB2 UET for the first time or if you are customizing a DB2 subsystem as the primary subsystem, select all tasks and steps.

   All of the required parameters have default values, which you can either accept or change.

   Tips:

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

3. Optional: Select other tasks and steps with a slash (/) and press Enter to activate the input fields. Either accept or change the default values that are displayed.
4. Press End to save your changes and exit, or issue the `SAVE` command to save your changes and stay on the Product Parameters panel.

Results

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

What to do next

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

Related tasks:

"Defining DB2 parameters"
DB2 parameters are parameters for a DB2 entry.

Defining DB2 parameters

DB2 parameters are parameters for a DB2 entry.

About this task

If you did not run the DB2 UET Discover EXEC, you must create and associate one or more DB2 entries before you can define the DB2 parameters. For more information, see "Creating and associating DB2 entries" on page 78.

Procedure

1. Specify `E` next to one or more DB2 entries in the associated list, which is in the Associated DB2 Entries and Parameter Status section on the Customizer Workplace panel, and press Enter. The DB2 Parameters panel is displayed, as shown in the following figure:

```
CCQPDB2                          DB2 Parameters                      15:56:08
Command ===>                                                 Scroll ===> PAGE
Ensure that values are specified for the required DB2 parameters. Press End to
save and exit.

Commands: SAVE VERIFYOFF

Product to Customize
  Product metadata library : RSQA.ABP220.DR110628.SABPD >  LPAR. . . : RS25
  Product name . . . . . : DB2 Utilities Enhancement  >  Version . : 2.2.0

                                                                      More:     +

 DB2 subsystem ID  . . . . . . . . . . . . : D91A
 Group attach name . . . . . . . . . . . . :
 *Is this DB2 subsystem the primary subsystem?
     YES (YES,NO)
 *Use DBCS with this SSID?  . . . . . . . . . NO  (YES,NO)
 *DB2 UET database name . . . . . . . . . . . ABP22DB
 *Drop DB2 UET database first?  . . . . . . . YES (YES,NO)
 *Free product packages and plans?. . . . . . YES (YES,NO)
 *Drop DB2 UET work database first? . . . . . YES (YES,NO)
 *DB2 UET v2.1 database name? . . . . . . . . ABP22DB

 General DB2 Information - common
  *Mode . . . . . . . . . . . . . . . . . . . NFM    (CM,CM8,CM9,NFM)
  *Level Number . . . . . . . . . . . . . . . 101 (810,910,101, 111)

 DB2 Libraries - common
  *Load Library . . . . . . . . . . . . . . . DSN.SDSNLOAD        >     Add...
  *Run Library  . . . . . . . . . . . . . . . DSN.RUNLIB.LOAD     >     Add...
  *Exit Library  . . . . . . . . . . . . . . .QA1A.SDSNEXIT       >     Add...

 DB2 Utilities - common
  *Plan name for the DSNTEP2 utility  . . . . DSNTEP2
```

Figure 11. The DB2 Parameters panel

2. Specify values for all parameters that are displayed.

   Tips:

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

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

   Tip: If you define a DB2 subsystem as the primary subsystem, the started task communicates with this subsystem when the task is started, and the subsystem is used to record audit and logging information about product activities in DB2 tables. Secondary subsystems are not used to record audit and logging activity from the started task. Therefore, the auditing and logging tables are not created

on secondary subsystems. However, other tables are created for secondary subsystems for other functions within DB2 UET, so the DB2 UET database and work database are still required.

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

Results

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

What to do next

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

Related tasks:

"Defining DB2 UET parameters" on page 80
DB2 UET parameters are specific to DB2 UET.

Generating customization jobs

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

Procedure

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

- If you want to generate customization jobs at the product level and for any associated DB2 entries, issue the GENERATEALL command, and press Enter.
- If you want to generate customization jobs for specific DB2 entries, select the DB2 entries by specifying the G line command against them, and press Enter. The available DB2 entries are in the associated list in the Associated DB2 Entries and Parameter Status section.

Important: Regenerating customization jobs will replace any existing jobs, including jobs that you might have manually modified after they were generated.

Results

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

What to do next

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

Tip: If the customization jobs are generated, but you are not ready to submit them, you can see them later by issuing the JOBLIST command on the Customizer

Workplace panel. The JOBLIST command displays the Finish Product Customization panel, which you can use to submit the jobs.

Submitting customization jobs

Submit the customization jobs to customize DB2 UET.

Before you begin

Ensure that the correct jobs are generated.

About this task

The following figure shows part of the Finish Product Customization panel. The table on this panel shows the customization jobs that are generated by Tools Customizer. They are grouped by job sequence number.

```
CCQPCST                  Finish Product Customization        Row 1 to 9 of 15
Command ===>                                                 Scroll ===> PAGE

Submit the members in the order in which they apply to all DB2 entries. To
submit the job, browse the member and issue the TSO SUBMIT command, or browse
the customized library and submit the jobs from there.

Product to Customize
  Product metadata library : RSQA.ABP220.DR110628.SABPD  > LPAR  . . : RS25
  Product name . . . . . . : DB2 Utilities Enhancement   > Version . : 2.2.0

Line Commands: E - Edit  B - Browse

   Product customization library . : CSJENN.TESTING2.$RS25$.ABP220           >

 Cmd Member   SSID GrpAttch Template Date       Description
 -   -------- ---- ----     -------- ---------- ------------------------------>
     A0OBJCAA D91A  --      ABPOBJCR 2011/10/13 Create DB2 UET DB2 objects
     A0OBJCAB QA1B  --      ABPOBJCR 2011/10/13 Create DB2 UET DB2 objects
     A1IDXCAA D91A  --      ABPIDXCR 2011/10/13 Create optional catalog indexes
     A1IDXCAB QA1B  --      ABPIDXCR 2011/10/13 Create optional catalog indexes
     A2MIGRAA D91A  --      ABPMIGRT 2011/10/13 LOG and AUDIT tables MIGRATION
     A3MIGRAA D91A  --      ABPMIGR2 2011/10/13 JOURNAL tables MIGRATION
     A4RSTCAA D91A  --      ABPRSTCR 2011/10/13 Create optional RUNSTATS job
     A4RSTCAB QA1B  --      ABPRSTCR 2011/10/13 Create optional RUNSTATS job
     A5BNDCAA D91A  --      ABPBNDCR 2011/10/13 Bind the DB2 UET plan
     A5BNDCAB QA1B  --      ABPBNDCR 2011/10/13 Bind the DB2 UET plan
     A6WKLCAA D91A  --      ABPWKLCR 2011/10/13 Create an optional SAMPLIB memb
     A6WKLCAB QA1B  --      ABPWKLCR 2011/10/13 Create an optional SAMPLIB memb
     A7OPTCAA D91A  --      ABPOPTCR 2011/10/13 Create the options module
     A8MODCAA D91A  --      ABPMODCR 2011/10/13 Create BCAN, BGLB, PROC, PLCY m
     A9SMPCAA D91A  --      ABPSMPCR 2011/10/13 Create maintenance members
     B0RUNCAA D91A  --      ABPRUNCR 2011/10/13 Create product CLISTs
```

Figure 12. The Finish Product Customization panel

The member-naming conventions depend on whether the customization jobs are for DB2 entries, and LPAR, or the product.

Customization jobs for DB2 entries

The members use the following naming convention:

<job_sequence_number><job_ID><DB2_entry_ID>

where

job_sequence_number

Two alphanumeric characters, A0 - Z9, that Tools Customizer

assigns to a customization job. The number for the first template in the sequence is A0, the number for the second template is A1, and so on.

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

DB2_entry_ID
Two alphanumeric characters, AA - 99, that Tools Customizer assigns to a DB2 entry.

For example, the XYZBND*DB2_entry_ID_1* and XYZBND*DB2_entry_ID_2* jobs are generated from the XYZBNDGR template, and the XYZ4*DB2_entry_ID_1* and XYZ4*DB2_entry_ID_2* jobs are generated from the XYZ4 template. If the jobs are generated on two DB2 entries, the following member names are listed sequentially: A0BNDGAA, A0BNDGAB, A14AA, A14AB.

Customization jobs for the product
The members use the following naming convention:

<*job_sequence_number*><*job_ID*>

where

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

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

For example, the XYZBNDGR job is generated from the XYZBNDGR template, and the XYZ4 job is generated from the XYZ4 template. The following member names are listed sequentially: A0BNDGR, A14.

Procedure

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

Results

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

Important: If you created a BIND job for a secondary subsystem, return code 4 is normal for that job.

What to do next

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

Browsing parameters

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

Procedure

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

Copying DB2 entries

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

About this task

Go to the step that applies to your environment:

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

Procedure

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

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

In some cases, the DB2 parameter values might contain the DB2 subsystem ID as an isolated qualifier in data set names. For example, in the DB01.DB01TEST.DB01.SANLLOAD, data set name, the DB01 subsystem ID is isolated in the first and third qualifiers but is not isolated in the second qualifier. When the DB2 subsystem ID is an isolated qualifier in data set names, the Change DB2 Subsystem ID in DB2 Parameter Values panel is displayed. Otherwise, the Customizer Workplace panel is displayed.

d. If the Change DB2 Subsystem ID in DB2 Parameter Values panel is displayed, specify an option for changing the subsystem IDs. Otherwise, skip this step.
   - To change the subsystem ID in isolated qualifiers in data set names, specify option 1, and press Enter.
   - To use the same subsystem ID in all values, specify option 2, and press Enter.

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

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

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

What to do next

Edit any of the parameters or generate the jobs.

Related concepts:

"Tools Customizer terminology" on page 441
Tools Customizer uses several unique terms that you should be familiar with before you begin to use Tools Customizer.

Removing DB2 entries

You can remove DB2 entries from the associated list.

About this task

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

Procedure

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

Related concepts:

"Tools Customizer terminology" on page 441
Tools Customizer uses several unique terms that you should be familiar with before you begin to use Tools Customizer.

Deleting DB2 entries

You can delete DB2 entries from the master list.

About this task

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

Procedure

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

What to do next

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

Displaying customization jobs

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

About this task

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

Procedure

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

Maintaining customization jobs

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

About this task

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

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

Procedure

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

Using Tools Customizer in a multiple-LPAR environment

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

About this task

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

This behavior occurs in the following conditions:

- Tools Customizer is installed on a DASD device that is shared by multiple LPARs.

- After a product is customized by using Tools Customizer, the data store is copied to another LPAR.

Procedure

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

Install one instance of Tools Customizer on one LPAR

If you intend to reuse the customization values for all the instances of your products on all LPARs, use this method.

1. Associate all the DB2 entries in this one instance of Tools Customizer. The LPARs on which the DB2 subsystems reside do not matter.
2. Generate the customization jobs for each DB2 entry.
3. Copy the generated customization jobs to the LPAR to run against the specific DB2 entries. Some LPAR-specific edits might be required. You can make these edits in the customized jobs that you copied. Note that this situation is one of the few situations where you might need to make manual changes to the jobs that are customized by Tools Customizer.

Install one instance of Tools Customizer on each LPAR

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

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

APF-authorizing the load library

Ensure that the product load library is APF-authorized so that it will be available for use. This customization step is required.

About this task

DB2 UET requires that the product load library (*hlq.mlq*.SABPLOAD) be authorized by the z/OS Authorized Program Facility (APF). To include this load library in your system APF-authorized list, issue the following z/OS operator command:

```
SETPROG APF,ADD,DSNAME=hlq.mlq.SABPLOAD,VOLUME=volser
```

Where

- *hlq* is the high-level qualifier.
- *mlq* is the mid-level qualifier that you specified when you ran Tools Customizer.
- *volser* is the volume serial number of the DASD device where the SABPLOAD library resides.

Copying the started task PROC

Copy the DB2 UET started task PROC to your system PROCLIB to make the started task address space available to the user interfaces for the product. This customization step is required.

About this task

Run the A6MODCR job located in the data set that is specified in the **Product Customization Library** field on the Tools Customizer Finish Product Customization panel. This job will create the *abpid*PROC member in your *hlq.mlq*.SABPSAMP library. This job was only created if you selected the Tools Customizer subtask **Create BCAN, BGLB, PROC, PLCY Members**.

Procedure

1. Copy the *abpid* PROC member that gets created in the *hlq.mlq*.SABPSAMP to a member in your system PROCLIB.
2. In the EXEC statement, ensure that `REGION=0M` is specified to avoid storage problems. Also, ensure that `TIME=1440` is specified to allow DB2 UET to run for an unlimited amount of time.
3. Ensure that the STEPLIB and SVCLIB data sets are APF-authorized.
4. If you plan to monitor threads across multiple DB2 subsystems that have different DB2 versions, ensure that the STEPLIB concatenation specifies the lowest (earliest) of these DB2 versions as the DSNLOAD library. Otherwise, connection problems might occur when you attempt to manage threads on DB2 subsystems other than the primary subsystem that contains the audit and logging tables.

What to do next

Copy the DSNUTILF module from the *hlq.mlq*.SABPLOAD product library to a STEPLIB or JOBLIB DD that is used in your DB2 utility jobs. See the task "Copying the DSNUTILF module."

Copying the DSNUTILF module

If you plan to use the DSNUTILB intercept, copy the DB2 UET DSNUTILF load module to a data set in the STEPLIB or JOBLIB concatenation for your DSNUTILB utility jobs.

About this task

This customization step is required for the DB2 UET started task to perform DSNUTILB interception services.

Procedure

1. Copy the DSNUTILF module into one of the APF-authorized libraries in the STEPLIB/JOBLIB concatenation.
2. Optionally, you can copy the DSNUTILF module to a data set other than RUNLIB.LOAD, SDSNEXIT or SDSNLOAD. If you choose this method, ensure that the data set is APF-authorized and is contained in the JOBLIB of your utility job or in the STEPLIB of the DSNUTILB jobsteps.
3. Depending on how you invoke DSNUTILB, you might also need to specify the target data set at the following locations:
   - If you plan to invoke DSNUTILB by using the DSNUTILS or DSNUTILU stored procedure, concatenate the DB2 UET LOAD library to the STEPLIB/JOBLIB concatenation.
   - If you plan to invoke DSNUTILB from a TSO or ISPF session, specify the target data set name in the STEPLIB or ISPLLIB concatenation for your TSO or ISPF sessions.

Results

Note that with the DSNUTILF module in the load library concatenation, the DSNUTILB program will still be able to operate normally even if the DB2 UET STC becomes unavailable or if the DB2 UET intercept policy prevents DSNUTILB interception.

What to do next

Optionally customize the DSNUTILB interception parameters. See task "Customizing DSNUTILB intercept parameters (optional)."

Customizing DSNUTILB intercept parameters (optional)

During customization with Tools Customizer, you set values for DSNUTILB intercept parameters on the Product Parameters panel with the required task **Create SABPSAMP members**. You can modify the cancel and global parameters that the DSNUTILB intercept uses for all thread blocking and cancelation activities.

About this task

Tools Customizer created a job called A6MODCR that creates the *abpid*BCAN and *abpid*BGLB members (where *abpid* is the value that you specified in the Tools Customizer field **DB2 UET Configuration Name**) in the *hlq*.*mlq*.SABPSAMP library for your started task configuration. DB2 UET provides default values for all parameters in these members.

If you omit a parameter from one of these members, DB2 UET uses the default value for that parameter. If you specify an invalid value for a parameter, any thread blocking and cancelation processing that the DSNUTILB intercept attempts fails.

Tip: You can edit these members while the started task is running and performing interception to correct any syntax errors or change parameter settings. However, for revisions to take effect for an intercepted DB2 utility, you must re-run the utility.

Procedure

1. Edit the values for any of the following parameters in the *abpid*BCAN member, or accept the default values. These parameters pertain to each cancel request. You cannot add comments to this member or specify additional parameters.

   CANCEL_TYPE BACKOUT | NOBACKOUT | NO_UR_CHECKING
   Specify the type of cancelation processing for DB2 UET to use for threads that contain uncommitted units of work. Valid values are as follows.

   Important: Do not specify NOBACKOUT or NO_UR_CHECKING unless necessary. These cancel types are not supported for threads on a DB2 subsystem other than the local subsystem to which a connection is established based on the <DB2SYSTEM> element value in the intercept policy. In a DB2 data sharing environment, the options do not apply to other subsystems in the data sharing group.

   - BACKOUT: (default) Specify this value to have DB2 back out (roll back) any uncommitted work that exists in the current unit of

recovery when a thread is canceled. Backout processing ensures that the data integrity of the updated DB2 objects is maintained. Any updates that occurred after the last commit operation are backed out.

- NOBACKOUT: Use this value with caution; data integrity problems are possible in certain situations. This value causes DB2 UET to check for any outstanding units of recovery just before canceling threads but prevents DB2 from performing backout processing. If DB2 UET finds an outstanding unit of recovery for a thread, the thread is *not* canceled and return code RC12 is issued. Depending on the **ON_FAILURE** parameter in member *abpid*BGLB, the cancel request either terminates or continues to the next thread. By default, the request continues and cancels the other threads. If DB2 UET does not find an outstanding unit of recovery, the thread is canceled. In this situation, data integrity problems can occur if an outstanding unit of recovery is created for a thread after DB2 UET has determined that no outstanding units of recovery exist for the thread and before the thread is canceled.
- NO_UR_CHECKING: Use this value with caution; data corruption in your database is likely in certain situations. This value causes threads to be canceled without backout processing or any checking for outstanding units of recovery before cancelation. All threads that match your thread-filtering criteria are canceled, even if outstanding units of recovery exist for them. This value is equivalent to using the DB2 NOBACKOUT option on the -CANCEL THREAD command. For more information about the DB2 NOBACKOUT option, see the *IBM DB2 for z/OS Command Reference*.

ESCALATE YES | NO

Specify whether DB2 UET issues the z/OS CANCEL command to cancel a thread when the DB2 -CANCEL THREAD command fails to do so. Valid values are as follows:

- YES: DB2 UET can issue the z/OS CANCEL command to terminate the TSO user, batch job, or started task from which the thread originated. This escalated cancelation processing is supported only for the following connection types: CAF, IMSDLIBT, RRSAF, and TSO. Also, in a DB2 data sharing environment, escalated cancelation is supported only for threads on a local subsystem that you specify in a DB2SYSTEM element of the intercept policy.
- NO: (default) DB2 UET issues the DB2 -CANCEL THREAD command but does not escalate cancelation processing to the z/OS CANCEL command when the DB2 -CANCEL THREAD command fails to terminate a thread. Any threads that were not canceled remain active.

Important: DB2 UET will not perform an escalated cancelation, even if you set the ESCALATE parameter to YES, if a thread has a connection type that is not supported for escalated cancelation processing or if the CANCEL_ESCALATION_ACTIVE option in the started task initialization options member is set to NO.

CHECK_THDTERM_RETRY_COUNT *integer*

After issuing the DB2 -CANCEL THREAD command, DB2 UET checks the status of the canceled threads to determine whether DB2 has actually terminated them. If some threads are not yet terminated, DB2 UET continues to check the thread status until all of the canceled threads are actually terminated or until the maximum retry count that

you set with this parameter is reached. If the maximum retry count is reached and a thread has still not been terminated, DB2 UET issues an error message with the appropriate return code. If you also specify YES for the option to escalate thread cancel, DB2 UET will then issue the z/OS CANCEL command to terminate the thread. Valid values are in the range 1 - 32767.

The default value is 20.

Tip: To define how frequently thread-status checking occurs, set the parameter CHECK_THDTERM_RETRY_INTERVAL.

CHECK_THDTERM_RETRY_INTERVAL *seconds*
Specify how often (in seconds) DB2 UET checks the status of the threads that have been canceled to determine whether they are actually terminated in DB2. Valid values are in the range 1 - 32767.

2. Edit the values for any of the following parameters in the *abpid*BGBL member, or accept the default values. These global control parameters pertain to all thread blocking and cancelation activities that the DSNUTILB intercept will perform. You cannot add other parameters to this member.

ESCAPE (*character*)
Specify the escape character that you want to use to delimit a wildcard character in an intercept policy rule for selecting threads for cancelation when that character is used as an actual part of the data value and not as a wildcard. If you do not specify the escape character immediately before such a character, DB2 treats the character as a wildcard when performing pattern matching during thread filtering. The default escape character is + (plus sign). This parameter is ignored in environments that use a double-byte character set (DBCS). For more information about escape expressions and pattern matching, see the *IBM DB2 for z/OS SQL Reference*.

EXEC_TYPE (CHECKPARMS | EXECUTE | SIMULATE)
Specify the execution mode for thread-cancelation processing. Valid values are as follows:

- CHECKPARMS: DB2 UET validates the syntax of all input parameters that are specified without actually blocking and canceling any threads. The DB2 utility will still run although no threads are canceled.
- EXECUTE: (default) Perform thread blocking and cancelation. All threads that are selected based on the EXCLUDE and INCLUDE rules that you define in the DSNUTILB intercept policy are blocked and canceled.
- SIMULATE: Perform a trial run of thread-cancelation processing for a utility. DB2 UET simulates the cancelation of threads that are selected based on the EXCLUDE and INCLUDE rules that you define in the DSNUTILB intercept policy. The processing flow, messaging, and reporting are the same as with EXECUTE mode, but no threads are actually blocked and canceled. Use this option to pre-check which threads will be canceled and to troubleshoot potential errors. The DB2 utility will still run although no threads are canceled.

ON_FAILURE (CONTINUE | TERMINATE [RESET_OBJECT_STATUS YES | NO])
Specify whether thread-cancelation processing continues or terminates when an error occurs. Valid values are as follows:

- CONTINUE: (default) Thread-cancelation processing continues when an error occurs.
- TERMINATE: The job terminates and no additional cancelation processing occurs.

For thread-cancel operations that include thread blocking, you can control whether DB2 UET reinstates the original statuses of the DB2 objects when cancelation processing fails or one or more of the canceled threads do not actually terminate. During thread blocking, DB2 UET changes the statuses of the DB2 objects to either RO (read only) or UT (only the utility for which threads are being canceled has access).

To control whether DB2 UET leaves the object statuses in this state after a cancel failure, add the following subparameter:

```
ON_FAILURE (TERMINATE RESET_OBJECT_STATUS YES|NO)
```

- YES: DB2 UET reinstates the original statuses of the objects.
- NO:DB2 UET retains the object statuses that were in effect during thread blocking.

REPORT_TYPE (SUMMARY|DETAIL)

Specify the level of detail in reports that DB2 UET provides for actual or simulated runs of thread-cancelation processing. Valid values are as follows:

- SUMMARY: (default) Print the Threads Canceled report and the Threads Canceled Unit of Recovery report for each cancel request for which active threads are selected for cancelation.
- DETAIL: Print the following reports for each cancel request:
  - Threads Canceled
  - Threads Canceled Unit of Recovery
  - All Active Threads
  - All Active Threads Unit of Recovery
  - All Active Threads Objects Referenced

  If no threads are selected for cancelation processing for a cancel request, DB2 UET generates only the reports on all active threads.

THREAD_QUIESCE_TIME (*seconds*)

For thread-blocking actions, specify the number of seconds that DB2 UET waits between initiating thread blocking and canceling the active threads. This delay allows applications that are using the existing threads to complete units of work or quiesce before thread cancelation. The default value 0 causes DB2 UET to cancel existing threads immediately after initiating thread blocking.

3. If you copied the *abpid*BCAN and *abpid*BGLB members under other names before editing them, make sure that you specify the new member names in the started task initialization options member (*abpid*OPTS). The ABPBMAIN_CANCEL_MEMBER and ABPBMAIN_GLOBAL_MEMBER options in the *abpid*OPTS member must correctly reference the *abpid*BCAN and *abpid*BGLB members for the started task to use these parameters for thread-cancelation processing for intercepted DB2 utilities.

What to do next

Optionally, make any necessary edits to the options module created by Tools Customizer. See task "Setting up additional initialization options members (optional)" on page 97.

Related concepts:

"Task flow for blocking and canceling threads" on page 183
This task flow presents the high-level tasks that you will need to perform to block and cancel threads by using the DSNUTILB intercept.

Setting up additional initialization options members (optional)

The initialization options member, called *abpid*OPTS, is used by the DB2 UET started task upon initialization. Tools Customizer customized *abpid*OPTS for the ABPID that you specified during customization. If you plan to run multiple started tasks to monitor different DB2 subsystems, you must create separate initialization options members for each. This customization step is required if you will use multiple started tasks.

About this task

Tools Customizer set values for many of these options based on the values that you specified in the customization panels.

You can temporarily override the values for the following options from the Control System panel in the ISPF interface. Any changes that you make from this panel are retained only until you restart the started task. The changes are not saved to the initialization options member.

- AUDIT
- ACTIVE
- LOGGING_ACTIVE
- TRACE_ACTIVE
- WORKFILE_DATACLAS
- WORKFILE_MGMTCLAS
- WORKFILE_STORCLAS
- WORKFILE_UNIT
- WTO_ROUTCDE
-

Procedure

1. Locate the *abpid*OPTS member that was generated for your started task configuration. The member resides in data set *hlq.mlq*.SABPSAMP. The member is identified in the started task ABPOPTS DD statement of the *abpid*PROC.
2. Edit option values as required for your site.

   Consider whether an option is required or optional. Some options are optional because they have default values. If you omit an "optional" option from the initialization options member, DB2 UET uses the default value for that option. The following options are required and must be present in the started task initialization options member:

   - ABPID
   - DB2_PLAN_NAME
   - DB2_SSID
   - WORK_DATABASE_NAME
   - SVC_NUMBER

Example

The following example shows the initialization options in XML. For more information about each option, see "Started task initialization options."

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE OPTIONS SYSTEM "DD:DTD(ABPDTDOP)">
<!-- ************************************************************ -->
<OPTIONS
 ABPBMAIN_CANCEL_MEMBER="abpidBCAN"
 ABPBMAIN_GLOBAL_MEMBER="abpidBGLB"
 ABPID="abpid"
 AUDIT_ACTIVE="YES"
 AUDIT_MAX_AGE="0"
 CANCEL_ESCALATION_ACTIVE="NO"
 COPY_FASTREP="PREFERRED"
 COPY_DEBUG="NONE"
 DB2_CONNECT_TO_ALL_SUBSYSTEMS="YES"
 DB2_CONNECTION_IDLE_TIMEOUT="300"
 DB2_PLAN_NAME="plan_name"
 DB2_SSID="subsystem_identifier"
 DB2_TASK_COUNT="2"
 DB2_TASK_IDLE_TIMEOUT="900"
 DSNUTILB_RETRY_COUNT="100"
 DYNAMIC_SYSOUT_CLASS="*"
 LOGGING_ACTIVE="YES"
 LOGGING_MAX_AGE="0"
 WORK_DATABASE_BUFFERPOOL="bufferpool_name"
 WORK_DATABASE_NAME="database_name"
 WORK_DATABASE_STOGROUP="stogroup_name"
 OVERRIDE_MAPPING_TABLE="value"
 POST_CANCEL_EXIT="NONE"
 PRE_CANCEL_EXIT="NONE"
 SECURITY_EXIT="NONE"
 SVC_NUMBER="svc_number"
 SHADOW_DBPRE="?ABPSHADPR?"
 SHADOW_SCHEMA="?ABPSHADSC?"
 TRACE_ACTIVE="YES"
 TRACE_SIZE="1"
 WORKFILE_DATACLAS="data_class"
 WORKFILE_MGMTCLAS="management_class"
 WORKFILE_STORCLAS="storage_class"
 WORKFILE_UNIT="unit_name"
 WORKLIST_ERROR_MAX_AGE="0"
 WTO_ROUTCDE="11"
/>
```

What to do next

If you would like to prevent users from being able to cancel threads using DB2 UET, you may create a security exit that is invoked each time a thread cancel command is issued. See task "Creating a security exit (optional)" on page 106.

Related reference:

"Started task initialization options"
Learn about the started task initialization options that you must set in the initialization options member during DB2 UET customization.

Started task initialization options

Learn about the started task initialization options that you must set in the initialization options member during DB2 UET customization.

Tools Customizer generates the sample initialization options member *abpid*OPTS (where *abpid* is the started task configuration ID that you specified in Tools Customizer) in the *hlq.mlq*.SABPSAMP library for your use. This member specifies the options with which your started task will be initialized. You can use the sample initialization options member as provided or edit it. Your customized initialization options member must reside in the partitioned data set (PDS or PDSE) that is allocated by the ABPOPTS DD statement of the started task PROC. For more information, see "Setting up additional initialization options members (optional)" on page 97.

Option descriptions

Most initialization options have default values. For the options for which variables are displayed, the Tools Customizer supplies values based on your input in the customization dialogs. If you type a value for an option in the initialization options member, make sure that you enclose your value in double quotation marks. All required options must be defined in the initialization options member. However, the optional options can be omitted.

If an option has a default value, and you specify spaces for the value or an empty value, then DB2 UET uses the default value for that option.

ABPBMAIN_GLOBAL_MEMBER="*member_name***" (optional)**
Specifies the name of the member that contains the optional cancel-control parameters for any thread cancelations that you perform by using the DSNUTILB intercept. These options are:

- CANCEL_TYPE
- ESCALATE
- CHECK_THDTERM _RETRY_COUNT
- CHECK_THDTERM _RETRY_INTERVAL

Tools Customizer generates the sample member *abpid*BCAN (where *abpid* is the started task configuration ID that you specified) in the *hlq.mlq*.SABPSAMP library for your use. If you change the name of this member, you must specify the new name in this option. The member must reside in the same partitioned data set (a PDS or PDSE) as the started task initialization options member (that is, the data set that is allocated by the ABPOPTS DD statement in the started task PROC).

ABPBMAIN_GLOBAL_MEMBER="*member_name***" (optional)**
Specifies the name of the member that contains the optional global parameters for any thread cancelations that you perform by using the DSNUTILB intercept. These options are:

- ESCAPE
- EXEC_TYPE
- ON_FAILURE, REPORT_TYPE
- THREAD_QUIESCE_TIME

When Tools Customizer ran, DB2 UET generated the sample member *abpid*BGLB (where *abpid* is the started task configuration ID that you specified) in the *hlq.mlq*.SABPSAMP library for your use. If you changed the name of this member, you must specify the new name in this option. This member must reside in the same PDS or PDSE as the started task initialization options member (that is, the data set that is allocated by the ABPOPTS DD statement in the started task PROC).

ABPID="*identifier***" (required)**
Specifies the identifier for the DB2 UET configuration on your z/OS system. No default value is provided. You specified this value when the Tools Customizer ran. The Tools Customizer used that ABPID value in the names of the members that it generated for your started task configuration (the *abpid*BCOMX, *abpid*BMAIN, *abpid*OPTS, *abpid*PLCY, and *abpid*PROC members). If you specify another ABPID value by editing this option, the names of these members will no longer correspond to the ABPID value; therefore, it might be more difficult to determine the started task configuration to which the members pertain.

AUDIT_ACTIVE="YES"|NO (optional)
Controls whether DB2 UET records audit information for thread cancelation activities in a DB2 table. Valid values are YES and NO (default). The product administrator can temporarily override this setting by specifying another value in the **Audit status** field on the Control System panel.

AUDIT_MAX_AGE="*number_of_days***" (optional)**
Indicates the maximum number of days to retain rows for audit information in the audit table (ABPAUDIT). This number of days is counted from the time that the rows are inserted into the table. When a row reaches this age limit, it is eligible for deletion. It is automatically deleted from the table the next time that a new row is inserted into the table. Valid values are in the range 0 - 32767. The default value 0 prevents the automatic deletion of old rows from the audit table. If you accept the default value, you must manually delete old rows from the audit table periodically to prevent the table from becoming too large.

Use the sample SQL that is provided in the SABPSAMP member ABPAUDD.

COPY_FASTREP="PREFERRED"|"REQUIRED"|"NONE" (optional)
Specifies whether the use of DFSMSdss fast replication (FlashCopy®) is preferred, required, or not to be used. The value specified for this option does not affect concurrent copy or virtual concurrent copy processing. Valid values are as follows:

- PREFERRED: (default) Specifies that you want to use a fast replication method, if possible. If fast replication cannot be used, DFSMSdss completes the operation using traditional data movement methods.
- REQUIRED: Specifies that fast replication must be used. DFSMSdss stops processing the current data set if fast replication cannot be used. However, DFSMSdss continues processing the rest of the data sets using fast replication. When the DEBUG(FRMSG(MIN|SUM|DTL)) keyword is not specified, DFSMSdss still issues summarized information regarding why a fast replication method cannot be used as though DEBUG(FRMSG(SUMMARIZED)) had been specified. The DEBUG(FRMSG(MIN | SUM | DTL)) keyword determines the amount of information provided for why you cannot use a fast replication method.
- NONE: Specifies that fast replication should not be used. DFSMSdss does not attempt to use fast replication and completes the operation using traditional data movement methods.

For more information about FASTREPLICATION, see the explanation of COPY command keywords in the IBM *z/OS DFSMSdss Storage Administration* documentation.

COPY_DEBUG=COPY_DEBUG="NONE" | "FRMSG" | "SMSMSG" (optional)
Specifies whether you want to use DEBUG as a diagnostic tool. Valid values are as follows:

- NONE: DEBUG is not used.
- FRMSG: DFSMSdss issues messages that explain why you cannot use fast replication or Preserve Mirror operation during COPY processing. Specify DEBUG(FRMSG) with one of the following sub-keywords:
  - FRMSG(MINIMAL): DFSMSdss issues a message with a minimal level of information.
  - FRMSG(SUMMARIZED): DFSMSdss issues an informational message with summary information.
  - FRMSG(DETAILED): DFSMSdss issues a message with detailed information When applicable, detailed information regarding ineligible volumes is provided in the message text.
- SMSMSG: DFSMSdss displays ACS WRITE statements to the job output.

For more information about DEBUG, see the explanation of COPY command keywords in the *IBM z/OS DFSMSdss Storage Administration* documentation.

CANCEL_ESCALATION_ACTIVE="YES" | "NO" (optional)
Controls whether DB2 UET users are allowed to perform escalated cancelations of threads. An escalated cancelation uses the z/OS cancel command to terminate the job, TSO user, or started task that is associated with the thread. Valid values are as follows:

- YES: Allow escalated cancelations
- NO: (default) Do not allow escalated cancelations (use only the DB2 -CANCEL THREAD command).

DB2_CONNECT_TO_ALL_SUBSYSTEMS="YES" | "NO" (optional)
Controls whether DB2 UET attempts to connect to all active DB2 subsystems on the z/OS system on which it is configured, or only to the DB2 subsystem that is specified in the DB2_SSID initialization option (the subsystem that contains audit and logging information). Valid values are as follows:

- YES: (default) DB2 UET attempts to connect to all active DB2 subsystems by default.
- NO: DB2 UET attempts to connect only to the primary subsystem that is specified in the DB2_SSID option. Only that subsystem is listed on the Set DB2 System panel in the ISPF interface.

  Note: If you specify `NO` and create a DSNUTILB intercept policy that specifies DB2 subsystems other than the primary subsystem, no intercept processing occur on those additional subsystems.

DB2_CONNECTION_IDLE_TIMEOUT="*seconds*" (optional)
Specifies the maximum amount of time (in seconds) that the DB2 connection for a DB2 UET task can have no activity. When this time limit is reached, the connection to DB2 closes. Valid values are in the range 0 - 32767. The default value is 300. If you specify 0, this timeout option is disabled and will not cause an inactive connection to close. This timeout option does not apply to the subtask for the DB2 UET connection to the DB2 subsystem that is specified by the DB2_SSID option.

DB2_PLAN_NAME="*name*" (required)
Specifies the name of the DB2 plan that the DB2 UET configuration uses to

access the DB2 tables where it stores information about its operations (for example, the audit and logging tables). You specified a plan name for the ?ABPPLAN? variable for the primary subsystem during customization. No default value is provided.

DB2_SSID="*ssid*" (required)
Specifies the valid subsystem identifier (SSID) for the DB2 subsystem that contains the DB2 tables for the DB2 UET audit and logging information. You specified this ID for the ?SSID? variable for the primary subsystem during customization. No default value is provided.

DB2_TASK_COUNT="*integer*" (optional)
Specifies the maximum number of z/OS tasks that DB2 UET can start for connection to a single DB2 subsystem. Valid values are in the range 1 - 2147483647. The default value is 2.

DB2_TASK_IDLE_TIMEOUT="*seconds*" (optional)
Specifies the maximum amount of time (in seconds) that a subtask for a DB2 UET connection to DB2 can remain inactive after the connection closes (that is, after the DB2_CONNECTION_IDLE_TIMEOUT limit has been met). When this time limit is reached, the subtask ends. Valid values are from 0 through 32,767. The default value is 900. If you specify 0, this timeout option is disabled and will not cause an inactive subtask to end. This timeout option does not apply to the subtask for the DB2 UET connection to the DB2 subsystem that is specified by the DB2_SSID option.

DYNAMIC_SYSOUT_CLASS="*class*" (optional)
Specifies a SYSOUT class for the SYSOUT data sets that DB2 UET dynamically allocates during DSNUTILB interception for the SYSPRINT output for a utility job. This value can be any valid one-character JES SYSOUT class. The default value is an asterisk (*), which indicates that DB2 UET should use the default SYSOUT class that is specified for the job, started task, or TSO session under which DSNUTILB is running. If you have an output management product that captures and deletes SYSOUT data sets automatically, make sure that you set this option to a SYSOUT class that your output management product will *not* delete. Otherwise, your output management product might attempt to delete the SYSOUT data sets that DB2 UET dynamically allocates, thereby causing DSNUTILB interception errors. If you specify a value other than the asterisk (*) character, note that the following dynamically allocated SYSOUT data sets will still use the default asterisk (*) class: the SYSOUT data sets that are dynamically allocated for the output from the batch interface (ABPBMAIN) and the ABPSORT data sets (which are used in sort processing for the DSNUTILB intercept).

Note: DYNAMIC_SYSOUT_CLASS="*class*" should be customized for JES3. Using the default value (*) is not recommended in a JES3 environment. Instead, in a JES3 environment, make sure that you set this option to a SYSOUT class that is defined with the HOLD=TSO parameter. If this option is set to a SYSOUT class that is not defined with the HOLD=TSO parameter, the DSNUTILB intercept will not be able to recombine SYSOUT files that are produced by the DB2 UET and the DSNUTILB utility. In this case, the SYSOUT will show up in the JES3 spool as multiple files. Some of the files will be named SYSPRINT and others will have a system-generated file name such as SYS*nnnn*.

LOGGING_ACTIVE="YES"|"NO" (optional)
Controls whether DB2 UET logs messages about product performance and

operations in its DB2 log table. Valid values are YES (default) and NO. The product administrator can temporarily override this setting by specifying another value in the **Logging status** field on the Control System panel.

LOGGING_MAX_AGE="*integer*" (optional)
Indicates the maximum number of days to retain rows for logged messages in the logging table (ABPLOG). This number of days is counted from the time that the rows are inserted into the table. When a row reaches this age limit, it is eligible for deletion. It is automatically deleted from the table the next time a new row is inserted into the table. Valid values are in the range 0 - 32767. The default value 0 prevents the automatic deletion of old rows from the logging table. If you accept the default value, you must manually delete old rows from the logging table periodically to prevent the table from becoming too large.

Use the sample SQL that is provided in the SABPSAMP member ABPLOGD.

OVERRIDE_MAPPING_TABLE="NO"|"YES" (optional)
Indicates whether to replace existing MAPPINGTABLE specifications that you manually defined for the DB2 REORG TABLESPACE utility with the MAPPINGTABLE specifications that DB2 UET automatically generates for the utility. Valid values are YES and NO (default).

Regardless of how you set this option, note that DB2 UET will always add a MAPPINGTABLE specification to a REORG TABLESPACE utility statement that includes SHRLEVEL CHANGE if that statement does not already contain an existing MAPPINGTABLE specification.

POST_CANCEL_EXIT="*exit_name*"|"NONE" (optional)
Specifies the name of the user exit that you optionally created for performing processing that you determined is necessary after thread cancelations. For example, you might create a post-cancel exit to notify users when thread-cancelation processing completes. If you are not using a post-cancel exit, specify `NONE` (default).

PRE_CANCEL_EXIT="*exit_name*"|"NONE" (optional)
Specifies the name of the user exit that you optionally created for performing some processing that you determined is necessary prior to thread cancelations. For example, you might create a pre-cancel exit to determine the DB2 objects that an application or utility will need to access. If you are not using a pre-cancel exit, specify "`NONE`" (the default value).

SECURITY_EXIT="*exit_name*"|"NONE" (optional)
Specifies the name of the user exit that you optionally created for verifying user authority to perform thread-management functions such as blocking and canceling threads and to access specific ISPF panels. If you are not using a security exit, specify "`NONE`" (the default value).

SVC_NUMBER="*integer*" (required)
(Required) Specifies the numeric identifier for the DB2 UET supervisor call (SVC). The SVC is dynamically installed when the started task starts and dynamically removed when the started task stops. It is required for the product interfaces to communicate with the started task. Valid values are from 200 through 255. No default value is provided.

SHADOW_DBPRE="?ABPSHADPR?" (optional)

Specifies a prefix for the names of shadow objects that are used with the IFDISCARDS and SHRELEVEL REFERENCE options in the LOAD utility. The prefix can be one to four characters, and must follow standard DB2

naming conventions for databases. For each invocation of the Load Prevalidate or LOAD REPLACE SHRLEVEL REFERENCE feature, the product generates a unique set of shadow object names by using the specified prefix and appending a numeric suffix.

SHADOW_SCHEMA="?ABPSHADSC?" (optional)

Specifies the schema (creator) to be used for the shadow objects that the product creates for use with the IFDISCARDS and SHRELEVEL REFERENCE options in the LOAD utility. When executing the Load Prevalidate or the LOAD REPLACE SHRLEVEL REFERENCE feature, the product creates a unique set of shadow objects using the specified shadow database prefix in the object names. The product uses the value that you specify for shadow schema as the schema name for the shadow objects that it creates.

TRACE_ACTIVE="YES"|"NO" (optional)
Controls whether DB2 UET collects trace information. Specify "YES" to enable tracing, or specify "NO" to disable tracing. The default value is "YES". A trace is a record of DB2 UET internal processing that is primarily used by Support for diagnosing a problem. The product administrator can temporarily override this setting by specifying another value in the **Trace status** field on the Control System panel.

TRACE_SIZE="*megabytes***" (optional)**
Specifies the size (in MB) of the table in which DB2 UET stores trace information. Valid values are from 1 - 2147483647. The default value is 1. A value of 0 will result in no trace table allocation. A trace is a record of internal processing that is primarily used by Support for diagnosing a problem.

WORK_DATABASE_BUFFERPOOL="*bufferpool_name***" (optional)**
Indicates the name of the buffer pool that you want to use for the work objects that are created for the extended functionality of the CHECK DATA utility. The default value is BP0.

WORK_DATABASE_NAME="*database_name***" (required)**
(Required) Indicates the name of the database in which you want to store the work objects that are created for the extended functionality of the CHECK DATA utility. This database name must be different from the one in which you store the DB2 UET audit and logging tables. When you customized DB2 UET using the Tools Customizer, you specified a work database name for the primary subsystem. The Tools Customizer included that name in the appropriate CREATE DATABASE statement within the A0OBJCAA DDL member that it generated for the primary subsystem and for each additional subsystem that you defined during customization. No default value is provided.

WORK_DATABASE_STOGROUP="*storage_group_name***" (optional)**
Indicates the unique name of the DB2 STOGROUP that you want to use for the work objects that are created for the extended functionality of the CHECK DATA utility. The default value is "SYSDEFLT" (the DB2 default storage group).

WORKFILE_DATACLAS="*data_class***"|"NONE" (optional)**
Specifies an SMS data class for the temporary DASD data sets that are allocated by DB2 UET. Specify a valid SMS data class name, or specify "NONE" (the default value). If NONE is specified, DB2 UET does not use a data class when allocating these data sets.

WORKFILE_MGMTCLAS="*management_class***" | "NONE" (optional)**
Specifies an SMS management class for the temporary DASD data sets that are allocated by DB2 UET. Specify a valid SMS management class name, or specify "NONE" (the default value). If NONE is specified, DB2 UET does not use a management class when allocating these data sets.

WORKFILE_STORCLAS="*storage_class***" | "NONE" (optional)**
Specifies an SMS storage class for the temporary DASD data sets that are allocated by DB2 UET. Specify a valid SMS storage class name, or specify "NONE" (the default value). If NONE is specified, DB2 UET does not use a storage class when allocating these data sets.

WORKFILE_UNIT="*unit_name***" | "NONE" | "SYSALLDA" (optional)**
Specifies a unit name for the location where the temporary DASD data sets that are allocated by DB2 UET are stored. Specify a valid unit name of a storage device or the value "NONE". As a valid unit name, you can specify the value "VIO" if VIO (virtual input/output) storage groups are supported on your system and you want the temporary data sets to reside entirely in paging storage to improve performance. The default value is SYSALLDA, which indicates any available DASD device. If NONE is specified, DB2 UET does not use this parameter to determine where to store the workfile data sets.

WORKLIST_ERROR_MAX_AGE="*integer***" (optional)**
Specifies the maximum number of days to retain rows in the DSNUTILB intercept worklist-error tables. A DSNUTILB intercept worklist contains the enhanced SYSIN information for a DB2 utility and can be used for restart purposes if a utility terminates. Worklist data is moved to worklist-error tables for diagnostic use by Customer Support in certain situations. After rows in the worklist-error tables reach the specified age limit, they are eligible for deletion. The next time a new row is inserted into a worklist-error table, the rows that meet the age limit are deleted. You can specify a value from 0 through 32,767 for this option. The default value is 0, which prevents the deletion of old rows from the worklist-error tables based on this option. If you accept the default value, you might need to manually delete old rows from these tables periodically to prevent the tables from becoming too large. Use the sample SQL that is provided in the SABPSAMP member ABPWKED.

WTO_ROUTCDE="*number***" (optional)**
Specifies the routing code for write-to-operator (WTO) messages regarding DB2 UET operations. Routing codes identify the z/OS console to which to send WTO messages and are defined when DB2 is installed. Valid values are from 1 through 28. The default value is 11. The product administrator will be able to temporarily override this value by entering another routing code on the Control System panel.

Related tasks:

| "Setting up additional initialization options members (optional)" on page 97
The initialization options member, called *abpid*OPTS, is used by the DB2 UET started task upon initialization. Tools Customizer customized *abpid*OPTS for the ABPID that you specified during customization. If you plan to run multiple started tasks to monitor different DB2 subsystems, you must create separate initialization options members for each. This customization step is required if you will use multiple started tasks.

Creating a security exit (optional)

If you want to control user access to product commands for intercept processing and to certain ISPF panels, you can create a security user exit. This customization step is optional.

Before you begin

Before you begin, coordinate with your security administrator to determine the appropriate security scheme for your DB2 UET users. Also, review the reference information on the security exit. See "Security exit" on page 451

About this task

The security exit must be written in assembler language. A sample exit (ABPXSE00) and corresponding DSECT (ABPAPISE) are provided in the *hlq*.*mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified when you ran Tools Customizer.

Procedure

1. Copy the sample exit ABPXSE00 and the DSECT ABPAPISE to the library where you want to store your security exit.

   Important: Do *not* change the copy of the DSECT ABPAPISE. Maintain the DSECT in its original state to ensure that it properly defines the layout of the memory for the API control block.
2. Edit the copy of the exit ABPXSE00, as needed, to implement your security scheme. The sample exit includes comments to guide you through this process. Depending on your security requirements, you might need to edit the exit significantly or make only minor changes.
3. Assemble and link the exit code to create an executable exit routine.
4. Ensure that the load module for the exit is available to the started task at runtime either by including it in the APF-authorized DB2 UET load library or by putting it in its own APF-authorized load library and then including that library in the STEPLIB concatenation.
5. Specify the exit name in the SECURITY_EXIT option of the started task initialization options member.

What to do next

Optionally, create pre- or post-cancel user exits to perform actions either before or after a thread is canceled. See "Creating a pre- or post-cancel exit (optional)" on page 107.

Related reference:

"Security exit" on page 451
You can create an optional security user exit to control user access to DB2 UET ISPF panels and thread-management functions. The exit will apply to a single DB2 UET configuration. If you do not create a security exit, all users will have access to all panels and thread-management functions.

Creating a pre- or post-cancel exit (optional)

If you need to perform additional processing prior to or after thread cancelations, you can create a pre-cancel exit, post-cancel exit, or both. This customization step is optional.

Before you begin

Before you begin, work with your production control coordinator, DB2 DBA, or other appropriate personnel at your site to determine your pre-cancel or post-cancel processing requirements. For example, you might need a pre-cancel exit to prevent the cancelation of threads that are running under a specific DB2 plan or a post-cancel exit to notify users when cancelations complete. Also, review the reference information on the pre-cancel and post-cancel exits. For more information about pre-cancel and post-cancel exits, see "Pre-cancel exit" on page 456 and "Post-cancel exit" on page 462

About this task

These exits must be written in assembler language. Sample exits and corresponding DSECTs are provided in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified when you ran Tools Customizer.

Procedure

1. Copy the sample exit and the DSECT for the pre- or post-cancel exit to the library where you want to store the exit.
   - For a pre-cancel exit, copy the sample exit ABPXPR00 and the DSECT ABPAPIPR.
   - For a post-cancel exit, copy the sample exit ABPXPO00 and the DSECT ABPAPIPO.

   Important: Do *not* change the copy of the DSECT. Maintain the DSECT in its original state to ensure that it properly defines the layout of the memory for the API control block.
2. Edit the copy of the sample exit, as needed, to implement your pre- or post-cancel processing requirements. The sample exit includes comments to guide you in this process. Depending on your requirements, you might need to significantly edit the exit or make only minor changes.
3. Assemble and link the exit code to create an executable exit routine.
4. Ensure that the load module for the exit is available to the started task at runtime either by including it in the APF-authorized DB2 UET load library or by putting it in its own APF-authorized load library and then including that library in the STEPLIB concatenation.
5. Specify the exit name in the PRE_CANCEL_EXIT option or POST_CANCEL_EXIT option of the started task initialization options member.

What to do next

Start the started task to begin using the product. See "Starting the started task" on page 109.

Related reference:

"Post-cancel exit" on page 462
You can create an optional post-cancel user exit to perform any processing that you consider to be necessary after canceling DB2 threads from the DB2 UET ISPF or batch interface. This exit applies to a single DB2 UET configuration.

"Pre-cancel exit" on page 456
You can create an optional pre-cancel user exit to perform any processing that you consider to be necessary prior to canceling DB2 threads from theDB2 UET ISPF or batch interface. The exit will apply to a single DB2 UET configuration.

Migrating existing data

You can use Tools Customizer to migrate data from your existing DB2 UET version.

About this task

You can migrate data from your DB2 UET AUDIT and LOG tables and your JOURNAL tables during Tools Customization. On the **Product Parameters** panel, complete all necessary fields for your customization.

Procedure

1. On the **Product Parameters** panel, select the **Create DB2 UET objects** field.

   This step results in the generation of job **A0OBJCAx**, where **x** will be **A** for your first SSID, **B** for your second SSID, and so on. This job, when submitted, will create your new version objects for the given customized SSID.
2. Select one or more of the following fields for the tables that you want to migrate:
   - To extract data from your existing LOG and AUDIT tables, and load the data into the new tables, select **Create LOG/AUDIT Migration job**.
   - To extract data from your existing JOURNAL tables, and load the data into the new tables, select **Create JOURNAL Migration job**.

   This step results in the generation of jobs **AnMIGRaa**, where **n** is a sequence number dependent on the number of jobs that you selected to generate, and **aa** are two alphabetic characters appended to the end of the job name.
3. Press **Enter**. The **CREATOR name** field for existing tables becomes available.
4. Enter the **CREATOR** values for your existing tables.

Chapter 4. Getting started

Review set-up tasks that you must perform before you can use the DB2 Utilities Enhancement Tool for the first time. The following table lists and describes each significant task.

The following table lists and describes each significant task.

Task	Link to detailed instructions	Status
Start the started task		
Before you can use the product, you must start the started task.	"Starting the started task"	
Start the ISPF interface and specify user settings		
Before you can use it to perform product functions, you must start the ISPF interface.	"Starting the ISPF interface" on page 110	
You can specify settings that are saved to your ISPF profile and used for your future DB2 UET ISPF sessions.	"Specifying user settings for ISPF sessions" on page 111	
Become familiar with and configure the DSNUTILB intercept component		
You must use the DSNUTILB intercept component to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use it to block and cancel threads.	"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122	

Starting the started task

Start the DB2 UET started task so that you can begin using the product interfaces to block and cancel threads or implement the DB2 utility enhancements. This customization step is required.

About this task

To start the started task, issue the following operator command from the z/OS console:

```
S abpstc
```

Where *abpstc* is the member name of the DB2 UET PROC in the system PROCLIB.

If you issue the command from SDSF instead, begin with a forward slash as follows:

```
/S abpstc
```

The Tools Customizer generated the started task name based on the value that you specified in the Tools Customizer field **DB2 UET Configuration Name**, and then inserted that name in the started task PROC. If you changed the started task name in the PROC, make sure that you use that new name in Start command.

What to do next

Start the ISPF interface to view all active threads on the system, or to cancel specific threads. See the task "Starting the ISPF interface."

Using the ISPF interface

Learn about using the DB2 UET ISPF interface.

From the ISPF interface, you can easily monitor thread activity, review detailed information about specific threads, and cancel threads. To find the threads that you are interested in, you can define numerous types of thread-filtering criteria. For example, to view or cancel threads that are accessing certain DB2 tables, you can create a filter that identifies only active threads on those tables. Also, you can perform some product administration tasks from the ISPF interface. For example, you can temporarily override a few started task initialization options.

Related concepts:

"Canceling threads by using the ISPF interface" on page 185
The DB2 UET ISPF interface provides a central point from which you can interactively monitor and cancel DB2 threads that originate from various sources in your environment.

Starting the ISPF interface

Start the ISPF interface so that you can use it to view or cancel DB2 threads, view the product status, or override selected started task initialization options.

About this task

If you copied the REXX EXEC for running the interface to another data set or data set member, make sure that you specify the name of that data set or member in the command statement.

The variables in the command statement are as follows:

- *hlq* is the is the high-level qualifier that you specified during customization.
- *mlq* is the is the mid-level qualifier that you specified during customization.

Procedure

1. Issue the following operator command from the z/OS console:
   ```
   TSO ex 'hlq.mlq.SABPCLST(ABPF)'
   ```
2. Press Enter.

Results

- If you are starting the interface for the first time, DB2 UET automatically displays the Set ABPID panel so that you can select the DB2 UET configuration to use. After you specify a configuration, DB2 UET automatically displays the Set DB2 System panel so that you can select the DB2 subsystem to which to connect.
- If you previously selected a product configuration and DB2 subsystem, DB2 UET displays the Main Menu first. If the selected product configuration is not running when you start the interface, the Set ABPID panel is displayed and lists any product configurations that are currently running. You can either select one of the active product configurations or ask the product administrator to start the product configuration that you usually use.

Related reference:

"Main menu" on page 114
The Main Menu is normally the first panel that is displayed after you start the DB2 UET ISPF interface. From the Main Menu, you can navigate to the other panels in the interface.

Specifying user settings for ISPF sessions

When you start using the DB2 UET ISPF interface, specify the user settings that you want to apply to your ISPF sessions. These settings are for your personal use only.

You must select the DB2 UET configuration ID and the DB2 subsystem to which you want DB2 UET to automatically connect whenever you start the interface. You can also set personal default values for two optional user settings: a wildcard escape character and an option that controls whether any filtering criteria that you specify are saved.

All of your user settings are saved to your ISPF profile and will be used for your future DB2 UET ISPF sessions until you change the settings. You can change your user settings at any time, if necessary.

Selecting the configuration and DB2 system for the first time

The first time you start the ISPF interface, you must select the DB2 UET started task configuration and DB2 system that you want to use for your ISPF sessions. Your selections are saved to your ISPF profile.

About this task

DB2 UET automatically displays the Set ABPID panel immediately after you invoke the ISPF interface for the first time. After you select a product configuration ID, DB2 UET automatically displays the Set DB2 System panel. You must complete both of these panels before you can navigate to other panels in the interface.

To select your DB2 UET configuration and DB2 system the first time:

Procedure

1. Start the ISPF interface. The Set ABPID panel is automatically displayed, as follows:

```
DB2 Utilities Enhancement Tool   Set ABPID                    12:03:22
Command ===>                                                Scroll ===> PAGE

Commands: REFRESH                                    ABPID . . . : ABP1
                                                     DB2 system. : DBP1
Line commands: S - Select                            User ID . . : PDABCD

                                                                 Row 1 of 2
C   ID   STC Name ASID Version
    ABP1 ABP01    03EE  02.20
    ABP2 ABP02    03F2  02.20
```

Figure 13. Set ABPID

   The panel lists all of the DB2 UET configurations that are running on your z/OS system. For each configuration, the panel displays the configuration ID, the started task name, the address space identifier, and the DB2 UET version.

2. Type S (Select) in the **C** column next to the ID of the DB2 UET configuration that you want to use, and press Enter. The selected ID is displayed in the **ABPID** field in the upper right-hand corner of the panel.
3. Press the F3 key to save the selected DB2 UET configuration ID to your ISPF profile. The Set DB2 System panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   Set DB2 System                      12:09:52
Command ===>                                                  Scroll ===> PAGE

Commands: REFRESH                                      ABPID . . . : ABP1
                                                       DB2 system. : DBP1
Line commands: S - Select                              User ID . . : PDABCD

                                                                Row 1 of 4
C   SSID     Version DS Group Status
    DBP1       9.1.0           ACTIVE
    DBT1       9.1.0
    DB8A      10.1.0  DB8GRP   ACTIVE
    DB8B      11.1.0  DB8GRP   ACTIVE
```

Figure 14. Set DB2 System panel

The panel lists the subsystem identifiers (SSIDs) for all of the DB2 subsystems that DB2 UET discovered in your z/OS environment. The subsystems that have the status of ACTIVE are the subsystems on which the DB2 plan for DB2 UET has been bound.

4. Type S (Select) in the **C** column next to the SSID of a DB2 subsystem that has the status of ACTIVE, and press Enter. This subsystem is the one to which DB2 UET will establish a connection whenever you start the ISPF interface. The SSID of the selected subsystem is displayed in the **DB2 system** field in the upper right-hand corner of the panel.

   Important: Make sure that you select an active DB2 system on which the DB2 plan for DB2 UET has been bound. Otherwise, the following SQL error will be issued in the started task log when the started task attempts to connect to that subsystem:

   ```
   Connection to DB2 failed. SSID=DBP1
    DSNT408I SQLCODE = -923, ERROR: CONNECTION NOT ESTABLISHED: DB2
             ACCESS, REASON 00E30301, TYPE 00000800, NAME ABCDPLAN
    DSNT418I SQLSTATE = 57015 SQLSTATE RETURN CODE
    DSNT415I SQLERRP = DSNAET03 SQL PROCEDURE DETECTING ERROR
   Disconnection from DB2 was successful. SSID=DBP1
   ```

5. Press F3 to save your user settings to your ISPF profile. These settings will remain in effect from session to session until you change them.

Results

The Main Menu is displayed.

What to do next

You can now view or cancel threads on the selected subsystem (including threads that originate from remote subsystems that communicate with the selected subsystem by means of DDF) and on any subsystems in the same DB2 data sharing group as the selected subsystem.

Changing the configuration or DB2 system

At any time, you can change the DB2 UET started task configuration or the DB2 system that you use for your DB2 UET ISPF interface sessions.

About this task

To change your DB2 UET configuration or DB2 system:

Procedure

1. From the Main Menu, choose option **0** (User Settings). The User Settings menu panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   User Settings                      12:15:06
Option ===>

   1  Set DB2 system                                  ABPID . . . : ABP1
   2  Set DB2 Utilities Enhancement Tool ID           DB2 system. : DBP1
   3  Set personal defaults                           User ID . . : PDABCD
```

Figure 15. User Settings panel

2. Choose one of the following options:
   - **1** (Set DB2 System) to change the DB2 subsystem
   - **2** (Set DB2 Utilities Enhancement Tool ID) to change the DB2 UET configuration ID

   Either the Set DB2 System panel or the Set ABPID panel is displayed.
3. Type S (Select) next to the SSID of an active DB2 subsystem or next to a DB2 UET configuration ID (whichever is listed), and press Enter.
4. Press F3 to save your selection to your ISPF profile. The next time you run the ISPF interface, DB2 UET will automatically connect to the new DB2 system or product configuration that you selected. Your selection will remain in effect until you change it.

Setting default values for optional user settings

You can optionally specify your own default values for the escape character that is used for delimiting wildcard characters when these characters are part of an actual data value (not used as a wildcard) and for an option that controls whether to automatically save the filtering criteria that you specify:

About this task

DB2 UET provides default values for both of these user settings. You can accept these default values or specify different values if appropriate. The default values that you specify will be for your personal use only and will remain in effect from session to session until you change them.

Procedure

1. From the Main Menu, choose option **0** (User Settings). The User Settings menu panel is displayed.
2. Choose option **3** (Set personal defaults). The Set Personal Defaults panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   Set Personal Defaults               12:22:15
Command ===>

                                                      ABPID . . . : ABP1
                                                      DB2 system. : DBP1
                                                      User ID . . : PDABCD

Wildcard escape. . . . . +         (Wildcard escape character)
Save filter criteria . . . NO      (Yes/No)
```

Figure 16. Set Personal Defaults panel

3. In the **Wildcard escape** field, type the escape character that you want to use to delimit a wildcard character in thread-filtering criteria when that character is used as an actual part of the data value and not as a wildcard. If you do not specify an escape character immediately before such a character, DB2 treats the character as a wildcard when performing pattern matching during thread filtering. (For more information about escape expressions and pattern matching, see the IBM *DB2 Version 10 for z/OS SQL Reference*.) The default escape character is the plus sign (+). DB2 UET uses this default character if you do not specify a value or if you specify one of the following characters: a blank symbol ␣, a percent sign (%), or an underscore (_). If your environment uses a double-byte character set (DBCS), the default value or any other value that you specify in this field will be ignored.
4. In the **Save filter criteria** field, specify **YES** if you want to have any filtering criteria that you specify automatically saved from session to session. If you want your filtering criteria to apply to the current session only, specify **NO** (the original default value) instead.
5. Press the F3 key. Your settings are saved to your ISPF profile

Main menu

The Main Menu is normally the first panel that is displayed after you start the DB2 UET ISPF interface. From the Main Menu, you can navigate to the other panels in the interface.

The following figure shows the Main Menu:

```
IBM DB2 Utilities Enhancement Tool   Main Menu                        11:39:52
Option ===>

   0  User settings                                ABPID . . . : ABP1
   1  Specify thread filter criteria               DB2 system. : DBP1
   2  Display and cancel threads                   User ID . . : PDABCD
   3  Administrator functions

   X  Exit



 ------------------------------------------------------------------------+
 IBM*
 ROCKET**
 Licensed Materials - Property of IBM
 5655-T58
  (C) Copyright Rocket Software, Inc. 2002, 2016 All Rights Reserved.
 *Trademark of International Business Machines
 ------------------------------------------------------------------------+
```

Figure 17. IBM DB2 Utilities Enhancement Tool Main Menu

Menu options

0 - User settings
Choose this option to specify the DB2 UET configuration and the DB2 subsystem to which to connect. You can also specify personal default values for a wildcard escape character and an option that controls whether any filtering criteria that you enter is saved. All of these settings are saved to your ISPF profile and will be used for each of your ISPF sessions until you change them.

1 - Specify thread filter criteria
Choose this option to specify filtering criteria for restricting the set of DB2 threads that are displayed.

2 - Display and cancel threads
Choose this option to display active DB2 threads (all active threads or those that meet your filtering criteria), to cancel threads, or to drill down to detailed information about a specific thread.

3 - Administrator functions
Choose this option to view information about the started task that is running, including the settings for started task initialization options. You can also override some initialization options.

X - Exit
Choose this option to exit the DB2 UET ISPF interface.

Read-only fields

ABPID
Displays the identifier for the DB2 UET configuration that you are using to view or cancel threads. You specify this ID in your user settings.

DB2 system
Displays the SSID of the DB2 subsystem to which you are connected. You specify this DB2 subsystem in your user settings.

User ID
Displays the user ID under which the interface is currently running. Usually, this ID is the user ID that you specified when you logged on to TSO.

Related tasks:

"Starting the ISPF interface" on page 110
Start the ISPF interface so that you can use it to view or cancel DB2 threads, view the product status, or override selected started task initialization options.

Scrolling panels

If all of the information on a panel cannot be displayed on your terminal screen at one time, you can scroll the panel information vertically and horizontally.

About this task

On long panels such as the Thread Summary Report, you can scroll vertically to view all of the panel contents. On multiple-column list panels, you can scroll horizontally to reposition the columns for easier viewing. You can also scroll horizontally on the Cancel Thread Information panel to view the entire text of the listed messages.

When the contents of a panel exceeds the vertical size of a panel, one of the following "More" indicators is displayed above the scrollable portion of the panel (on the right side) to alert you that more information exists:

- `More: +` if more information exists below the current position
- `More: -` if more information exists above the current position
- `More: + -` if more information exists below and above the current position

To scroll a panel:

- To scroll vertically on a long panel, press either the F8 (Fwd) key to scroll downward or the F7 (Bkwd) key to scroll upward.
- To scroll horizontally on list panels, including the Cancel Thread Information panel, press either the F11 key to scroll right or the F10 key to scroll left.

Tip: If the **Scroll** field is displayed at the top right corner of a panel, you can control the scroll amount by specifying one of the following options in the field: `PAGE` (scroll one full page, where a page is the amount of information that is visible on the panel), `HALF` (scroll half a page), `MAX` (scroll to the top or bottom), `CSR` (scroll so that the current cursor position is at the top or bottom), and `DATA` (scroll to the point that is one line or character less than a full page).

Scrolling and expanding fields

If the fields on the Thread Filter Criteria panel contain values that are longer than the displayed entry lines, you can scroll and expand the fields so that you can view, add, edit, or delete the long values.

About this task

When you expand a field, a pop-window is displayed that contains multiple entry lines on which you can add, edit, or delete a long value in its entirety. You must expand a field to add a value that is longer than the entry line on the base panel. After you add a long value, you can scroll through it laterally from the base panel. A field that is expandable and scrollable is indicated by the greater than sign (>) or less than sign (<) to the right of its entry line.

To scroll and expand fields:

- To expand a field into a separate pop-up window, move your cursor to the field and press F6 (Expand). If you edit the field value, press F3 to save your changes and return to the Specify Thread Filter Criteria panel.
- To scroll to the left or to the right in a field, place the cursor in the field and use the following function keys:
  - If the > sign is displayed, press F11 to scroll to the right.
  - If the < sign is displayed, press F10 to scroll to the left.
  - If both the > and < signs are displayed, you can use the F11 and F10 keys to scroll in either direction.

Important: If a blank appears within the values that you specified by using the expand feature, apply the IBM APAR OA10828 for ISPF version 4.0 to correct this problem.

Sorting selection lists

On panels that contain a selection list with multiple columns, you can sort the list by a particular column to find an item more easily. This feature is useful on the following panels: the Set ABPID panel, the Set DB2 System panel, and the Thread Summary Report panel.

About this task

At the command line, specify the SORT primary command followed by either the column name or column number (the number that represents the order of the column in the left-to-right sequence of columns, excluding the **C** column). Use the following syntax:

SORT *column_name*|*column_number* A|D

Where the letter A stands for ascending order and the letter D stands for descending order.

For example, if you specify SORT Req D or SORT 5 D on the Thread Summary Report panel, DB2 UET sorts the **Req** column (showing the number of DB2 requests) in descending order and reorders the list accordingly, as shown in the following figure:

```
DB2 Utilities Enhancement Tool   Thread Summary Report                 11:57:53
Command ===>                                                  Scroll ===> PAGE

Commands: FILTER  REFRESH                            ABPID . . . . . : ABP1
                                                     DB2 system. . . : DBP1
Line commands: C - Cancel  S - Detail  O - Objects   User ID . . . . : PDABCD
               K - Escalated Cancel    I - Info
                                                      Thread filter . : NO

                                                                    Row 1 of 4
 C   Msg Name     St A   Req ID           AUTHID   Plan     ASID Token   Member
         DB2CALL  T       25 DB2CALL      PDUSRZ   PGMBPLAN 00B8     454
         DB2CALL  T       16 DB2CALL      PDUSRX   PGMCPLAN 00AC     429
         DB2CALL  T        8 DB2CALL      PDUSR1   PGMAPLAN 00CB     484
         DB2CALL  T  *     4 DB2CALL      PDUSR1   PGMAPLAN 00C8     485
```

Figure 18. Example of sorting a selection list

Tip: Alternatively, you can use the FIND command followed by a text string to find an item in a selection list, for example, FIND PGMBPLAN.

Selecting menu options and list items

Learn how to select an option from a menu panel or an item from a selection list in the DB2 UET ISPF interface.

About this task

Use these methods:

- To select a menu option, type the number that corresponds to the option and press Enter. You can use this method on the Main Menu, the User Settings menu, and the Administrative Functions menu. For example, on the Main Menu, you can type 1 and press Enter to display the Thread Filter Criteria panel.
- To select an item from a list , type S (Select) or a forward slash (/) in the **C** column next to the item, and press Enter. You can use this method on the following list-type panels: the Set ABPID panel, the Set DB2 System panel, and the Thread Summary Report panel.

Primary commands

You can specify primary commands at the command line on many DB2 UET ISPF panels to perform various functions.

Some commands perform functions that are product-specific, for example, the FILTER command on the Thread Summary Report panel. Other commands are standard for ISPF panels, for example, the REFRESH and CANCEL commands. The key primary commands that are available on a panel are listed on the top of the panel. You can press F1 to display Help information about the commands.

Command descriptions

DB2 UET supports the following primary commands on one or more of the ISPF panels:

CANCEL
On all panels, you can specify the `CANCEL` command to exit the panel without saving any changes that you made or implementing a menu choice.

FILTER
On the Thread Summary Report panel, you can specify the `FILTER` command to display the Thread Filter Criteria panel for filtering the threads that are listed.

FIND *text_string*
On panels that have selection lists (the Set DB2 System panel, Set ABPID panel, and Thread Summary Report panel), you can specify the `FIND` command followed by a space and a text string to locate an item on the panel. For example, if you specify `FIND D7B`, the cursor would jump to the first occurrence of "D7B" in the list.

HELP On all panels, you can specify the `HELP` command to display Help information for the entire panel or for a particular field. To display panel-level Help, type `HELP` at the command line and press Enter. To display field-level Help, type `HELP` at the command line, move your cursor to a field, and then press Enter. Alternatively, you can press the F1 key to display Help information.

Tip: The panel-level Help includes information about the product-specific primary commands that a panel supports.

INFO On the Display Thread Detail panel for a canceled thread, you can specify the `INFO` command to display the message log from cancelation processing.

REFRESH
On panels that display data for your environment (for example, threads, started task attributes, or DB2 resources), you can specify the `REFRESH` command to retrieve the latest data. DB2 UET re-reads the data from the started task and displays it for viewing. For example, after canceling a thread from the Thread Summary Report panel, you can specify the `REFRESH` command to determine if the thread has actually terminated; if the thread no longer appears in the list, it has terminated. Alternatively, you can press the F5 key to refresh a panel.

RESET
On the Thread Filter Criteria panel, you can specify the `RESET` command to clear the existing values from all of the entry fields so that you can specify new filtering criteria or remove the filter.

SAVE On the Control System panel, you must type `SAVE` at the command line and press Enter to save any changes that you made to the started task initialization options. If you press F3 only, your changes will not be saved.

SORT *column_number*|*column_name* **A**|**D**
On panels that contain a multi-column selection list, you can specify the `SORT` command to sort the list by a single column. Type `SORT` followed by the column name or the column number (the number in the left-to-right sequence of columns, excluding the **C** column), and then type either the letter `A` (for ascending order) or `D` (for descending order). When you press Enter, DB2 UET sorts the data in the specified column and reorders the list items accordingly. For example, if you specify `SORT 1 A` on the Set DB2 System panel, the list is sorted in ascending order based on the **SSID** column (the first column after the **C** column).

Displaying online Help

If you need assistance while working on a DB2 UET panel, you can display online Help information for the panel as a whole or for an individual field.

About this task

Panel-level Help is available for all panels. Field-level Help is available for only the following panels: the Thread Filter Criteria panel, the Display Thread Detail panel, the Display System Status panel, and the Control System panel.

To display panel-level Help, open a panel and then press the F1 key or type `HELP` at the command line and press Enter. If field Help is displayed instead, move the cursor outside of the field (the entry line or data value) and press F1.

To display field-level Help, open one of the panels that support field-level Help, move the cursor to the entry line or data value for a field, and then press F1. If you press F1 a second time, the panel-level Help information is displayed.

Using the batch interface

With the batch interface, you can manually create a job step for canceling threads and include it within a batch job.

This strategy is useful when the batch job runs utilities or applications that require access to certain DB2 objects. By placing the thread-cancelation job step before the utility or application job step, you can free the DB2 objects that the utility or application needs. If you have a utility or application that requires exclusive access to DB2 objects, you can create a thread-cancelation job step that also blocks new threads from forming on the DB2 objects until after the existing threads are canceled and the utility or application completes.

The batch interface cancels threads by using the DB2 -CANCEL THREAD command. You can specify the optional ESCALATE parameter to perform an escalated cancelation. For an escalated cancelation, DB2 UET first issues the DB2 -CANCEL THREAD command. Only if that command fails to terminate a thread doesDB2 UET then issue the z/OS Cancel command to terminate the TSO user, started task, or batch job that is associated with the thread.

The batch interface supports all of the same thread-filtering criteria as the ISPF interface plus a parameter for filtering threads based on a thread token value. Also, the batch interface supports parameters for controlling how often and how many

times DB2 UET checks the termination status of a thread for which a cancel command was issued. The job output will include messages that indicate the checking activity and termination status for each thread.

You can perform the following tasks using the batch interface:

- Create a job step that contains multiple CANCEL_THREADS requests (also called *cancel requests*), each with a different set of thread-filtering criteria.
- Block new threads from forming on the DB2 objects that a utility or application needs to access until after cancelation processing completes and the utility runs.
- Run a thread-cancelation job in simulation mode to check that the correct threads will be canceled when you actually run the job, or run the job in a mode that checks the parameter syntax only.
- View detailed reports that DB2 UET generates for each CANCEL_THREADS request during an actual or simulated run of a thread-cancelation job.
- Create a thread-cancelation job that uses DB2 NOBACKOUT processing, if necessary, instead of the default BACKOUT processing. (This type of thread cancelation should be performed only in exceptional circumstances because it can result in data integrity problems.)

Also, the batch interface is called by the DSNUTILB interface to perform any intercept processing that you configured in the intercept policy for DB2 utilities.

Review the following conceptual topics for more information about the features that are available for batch thread-cancel processing.

Related concepts:

"Blocking and canceling threads by using the batch interface" on page 208
The DB2 UET batch interface enables you to create a batch job step that cancels threads on the DB2 objects that an application or utility needs to access during batch processing. Optionally, this job step can also block new threads from forming on the objects that a utility needs until after the utility completes.

Blocking threads

You can create a batch job step that both cancels active threads on the DB2 objects that your batch utilities or applications need to access and blocks new threads from updating these objects until after the utilities or applications complete.

By incorporating thread-blocking into your batch jobs, you can ensure that your batch utilities and applications will have exclusive access to the DB2 objects that they need. Any other utilities and applications that are running will not be able to create new threads on these DB2 objects from the time when existing threads are canceled until after your batch utility or application completes.

DB2 UET blocks new threads by changing the status of the DB2 objects to one of the following values:

RO The utilities and applications that are running in your environment have read-only access to the DB2 objects. None of them can update the objects in any way (add, change, or remove data).

UT Only the DB2 online utilities can access to the DB2 objects. No applications can access the objects to either read or update data.

DB2 UET selects one of these statuses for a DB2 object based on the ACCESS parameter value (ALL, READ, or UPDATE) that is specified for the batch

thread-cancel job step. If the ACCESS value is ALL (the default value) or READ, DB2 UET changes the object status to UT. If the ACCESS value UPDATE, DB2 UET changes the object status to RO.

You must create a second DB2 UET job step to allow new threads to form on the DB2 objects once again. You run this job step after the thread-blocker job step and after the job step for the utility for which you are canceling threads. This "allow threads" job step restores the original statuses of the objects. It also removes all records for the uniquely identified thread-blocking operation from the DB2 table in which DB2 UET stores thread-blocker metadata. This table must be located on the DB2 subsystem on which threads were blocked or on a subsystem within the same DB2 data sharing group as that subsystem.

Tip: If a cancel request within a thread-blocker job step fails or if one or more of the canceled threads do not actually terminate, the new object statuses might remain in effect. You can control whether the new statuses remain in effect or the original statuses are reinstated by setting the optional ON_FAILURE (TERMINATE RESET_OBJECT_STATUS YES|NO) parameter in the thread-blocker job step.

Alternative run modes

You can run a thread-cancelation job in simulation mode or parameter-checking mode to verify your job setup before actually running the job in execution mode.

You set the run mode with the EXEC_TYPE global parameter. The following run modes are supported:

- CHECKPARMS only validates the input parameters. No simulated or actual thread-cancelation processing is performed.
- SIMULATE performs a trial run of the job so that you can check that your thread-filtering criteria selects the correct set of threads to cancel and troubleshoot any potential problems.DB2 UET displays the processing flow, messages, and reports for all cancel requests in the job but does not actually cancel any threads. No thread blocker activity is reported in the simulated output.
- EXECUTE actually runs the job. Threads that match the specified thread-filtering parameters are canceled.

The EXECUTE mode is used by default if you do not specify an EXEC_TYPE value.

Reporting

The DB2 UET batch interface generates reports for thread-cancelation jobs. You can view these reports to learn about the threads that were canceled and the threads that were active when the thread-cancelation job ran.

DB2 UET can generate up to five reports for each cancel request in a thread-cancelation job: two summary reports on canceled threads and three detail reports on all active threads. The reports are generated for either an actual run or a simulated run of a thread-cancelation job (when the EXEC_TYPE global parameter is set to either EXECUTE or SIMULATE). You can access the reports from the job output by using SDSF or an equivalent tool.

To control whether all reports are generated or only those on threads canceled, specify a value for the REPORT_TYPE global parameter in the thread-cancelation job. If you specify SUMMARY for this parameter, only the two summary reports

on threads canceled are generated, provided that some threads matched your thread-selection criteria. If you specify DETAIL, the three detail reports on all active threads are also generated.

If you do not include the REPORT_TYPE parameter, DB2 UET generates only the summary reports on threads canceled by default.

For descriptions of the reports, see "Reports" on page 236.

Related concepts:

"Reports" on page 236
DB2 UET generates reports for each cancel request in a thread-cancelation job. The reports contain information about the threads that were canceled and the threads that were active when the job ran. Print and keep these reports for your records.

Return code use

You can use the return codes that the DB2 UET batch interface returns for a thread-cancelation job step to control whether subsequent applications and programs in the batch job run.

The DB2 UET batch interface issues several different return codes for thread blocking and cancelation processing. You can use these codes to control subsequent batch processing. For example, you might want to prevent the execution of subsequent programs when return code 28 is issued. This return code indicates that a thread blocking or cancelation operation failed. If you allowed the subsequent programs to run, they might not be able to access the DB2 objects that they need because threads on those objects were not canceled.

For descriptions of the return codes, see "DB2 Utilities Enhancement Tool codes" on page 436.

Related reference:

"DB2 Utilities Enhancement Tool codes" on page 436
Review the descriptions of the return codes and abend codes that can be issued from the DB2 Utilities Enhancement Tool started task, DSNUTILB intercept, or batch interface to determine how DB2 Utilities Enhancement Tool processing ended. For descriptions of any DB2 codes that might be returned for DB2 Utilities Enhancement Tool processing, see the *DB2 for z/OS Codes* manual.

Preparing to intercept the DSNUTILB program and DB2 utilities

The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

To use the DB2 UET DSNUTILB intercept, you must perform the following configuration tasks:

- Create a DSNUTILB intercept policy in XML and specify the policy member in the started task PROC.

  The policy specifies the DB2 subsystem or subsystems for which to perform DSNUTILB interception and optionally specifies rules that control the blocking and canceling of threads on DB2 objects. For more information, see "Defining and using a DSNUTILB intercept policy" on page 129.

- Make sure that the started task initialization options that pertain to mapping tables and the intercept worklist-error tables are set appropriately for your environment and intercept processing needs.

After you perform these configuration tasks, the DSNUTILB intercept component can intercept the DSNUTILB program and analyze the DSNUTILB SYSIN stream for a utility job. The intercept divides the original SYSIN stream into separate worklist steps, each including a single utility command (for example, LOAD) and any applicable set-up statements (for example, LISTDEF, TEMPLATE, or OPTIONS). DB2 UET examines the worklist steps and the DSNUTILB intercept policy that you created to implement the enhanced processing for the DB2 utilities.

From time to time, you might need to perform some intercept management tasks. For example, you might to need to terminate a utility for which interception has occurred in a manner that removes the associated worklist data.

Related concepts:

"Defining and using a DSNUTILB intercept policy" on page 129
If you want to use the DSNUTILB intercept component to implement the DB2 UET enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads automatically for certain DB2 utility operations, you must define a DSNUTILB intercept policy.

"Managing DSNUTILB interception" on page 309
You can manage DSNUTILB interception by performing some routine and occasional tasks.

Chapter 8, "Manipulating data in input records before loading (LOAD utility enhancements)," on page 267
DB2 UET provides several options for the DB2 LOAD utility to enhance load processing. These options are in addition to those that the native DB2 LOAD utility provides. They are implemented only through the use of the DB2 UET DSNUTILB intercept interface.

Chapter 9, "Automatically creating mapping tables (REORG TABLESPACE utility enhancements)," on page 291
DB2 UET can automatically size and create the mapping table and mapping-table index that are required for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option. When the reorganization completes, these objects are automatically dropped to preserve space.

"Blocking and canceling threads by using the DSNUTILB intercept" on page 180
You can use the DSNUTILB intercept to transparently block and cancel threads for the DB2 online utilities. The intercept can block and cancel threads for all DB2 utilities for which this function is appropriate.

Considerations for using the DSNUTILB intercept component

Review these operational considerations if you plan to use the DSNUTILB intercept to implement the DB2 utility enhancements or to block and cancel threads.

- For intercept processing, the DSNUTILB intercept invokes the DB2 UET batch interface. The DB2 utility output will include "ABPB" messages that are issued by the batch interface and the batch reports on threads canceled and optionally on all active threads.
- If the original DSNUTILB SYSIN stream contains a LISTDEF set-up statement that defines a list of DB2 objects for a utility to process, you must define rules in

the intercept policy that will select these same objects for intercept processing. Unless the policy also identifies the LISTDEF objects, no thread-cancelation processing will occur for them.

- When the DB2 UET started task is not running, the DSNUTILB intercept does not intercept the DSNUTILB program and no intercept processing occurs. The DSNUTILB program runs as if the DSNUTILB intercept is not present.
- DB2 UET supports the DB2-supplied stored procedures DSNUTILS and DSNUTILU for invoking DSNUTILB from user applications. When DSNUTILB is invoked by one of these stored procedures, DB2 UET intercepts the SYSIN data that the stored procedure attempts to pass to DSNUTILB and implements the enhanced intercept processing that you configured.
- If you are running multiple DB2 UET started tasks to perform intercept processing on different DB2 subsystems, you must specify a DSNUTILB intercept policy for each started task. Any given DB2 subsystem can be intercepted by only one started task at a time. For more information, see "Considerations for running multiple started tasks" on page 32.
- When executing a COPY utility with the CONCURRENT keyword, DSNUTILB requires that a DSSPRINT DD statement be included if the SYSPRINT DD is allocated to a data set. DFSMS messages with a prefix of "ADR" are written to DSSPRINT in this case. This requirement is waived if SYSPRINT is allocated to SYSOUT (SYSOUT)=<class>. In this case, both the "DSNU" and "ADR" messages are written to SYSPRINT.

  However, when the above-mentioned scenario involves the DB2 UET DSNUTILB Intercept, DSSPRINT is not required regardless of the SYSPRINT allocation. The "ABP", "DSNU" and "ADR" messages are all written to SYSPRINT regardless of whether it is allocated to a JES SYSOUT class or a data set. If DSSPRINT is specified in the JCL when using the DB2 UET DSNUTILB Intercept, it will be ignored.

  Note: The DSNUTILB Intercept does not support allocations to data sets residing on tape for SYSIN or SYSPRINT DD statements in the utility job.

DSNUTILB intercept worklists

DB2 UET divides the DSNUTILB SYSIN syntax stream for a DB2 utility into separate *worklist steps*. Each worklist step is composed of a single utility command (for example, COPY) and any related setup statements (for example, OPTIONS, TEMPLATE, and LISTDEF). The entire set of worklist steps that is generated from a SYSIN stream comprises a worklist.

The division of the original SYSIN into worklist steps enables DB2 UET to block and cancel threads on DB2 objects shortly before a utility needs to access the objects, thereby reducing disruption to other DB2 users. Also, if a utility terminates abnormally and then is restarted by DB2, DB2 UET can use the worklist to resume intercept processing from the point of failure.

Tip: If you need to terminate a DB2 utility for which interception has occurred, it is recommended that you use the ABPMAINT utility with the TERM_UTILITY option rather than terminate the utility by using the DB2 -TERM UTILITY command.

The following figure shows how an original SYSIN stream that contains three REORG TABLESPACE commands is split into three worklist steps, each with the appropriate OPTIONS, LISTDEF, and TEMPLATE statements.

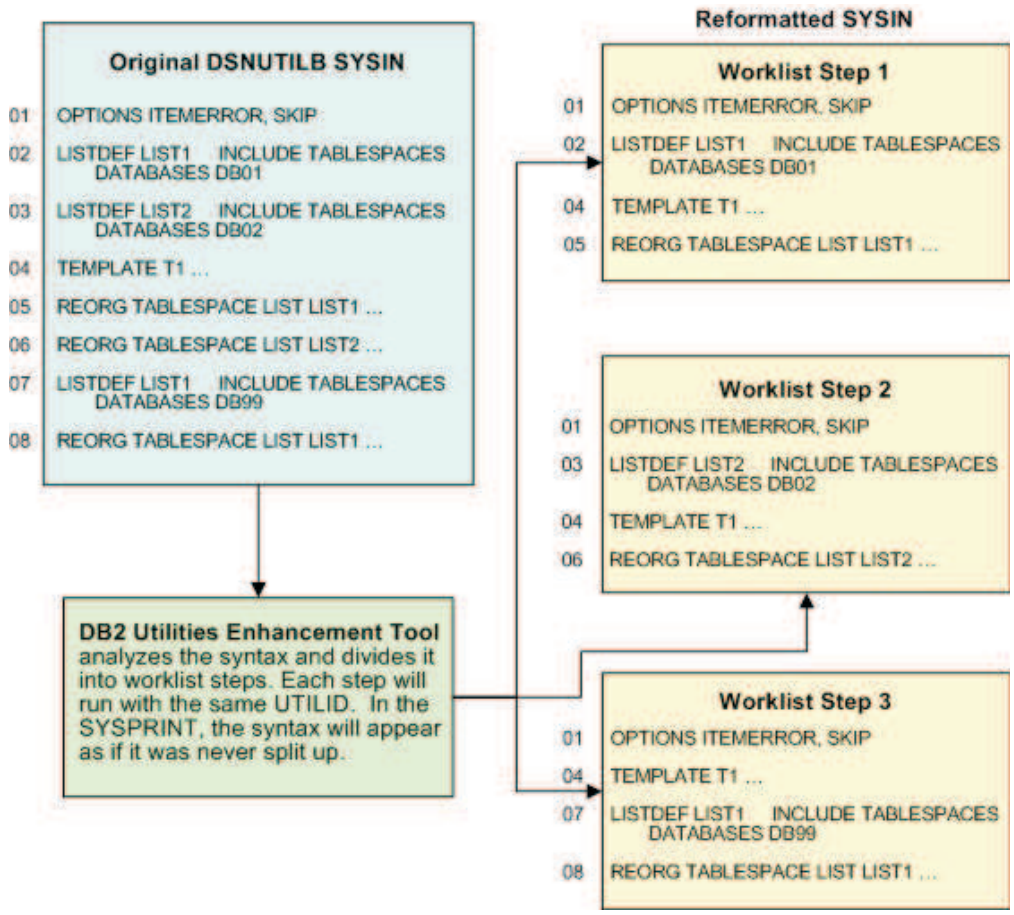


Figure 19. Division of DSNUTILB syntax into worklist steps

In this example, each LISTDEF list is matched with the appropriate REORG TABLESPACE command. Also, the OPTIONS and TEMPLATE statements, which pertain to multiple utility invocations, are correctly applied to all REORG TABLESPACE commands.

Restriction: When the original SYSIN syntax specifies a LISTDEF, all DB2 objects in the LISTDEF list must also be defined in the DSNUTILB intercept policy for thread-cancelation processing to occur for these objects.

DB2 UET stores data for a worklist across several DB2 tables. These *worklist tables* are created during customization on the primary subsystem and on each "additional" subsystem that you define. A corresponding set of *worklist-error tables* is also created during customization. Worklist data is moved to the worklist-error tables for diagnostic use by Customer Support in certain situations.

Worklist rows that are associated with a particular utility ID are automatically deleted from the worklist tables when intercept processing for that utility ID completes normally. Any worklist data that was moved to the worklist-error tables is deleted based on the age limit that you set in the WORKLIST_ERROR_MAX_AGE option of the started task initialization options member. If you specified 0 for this option, you can manually delete data from the worklist-error tables by using the SQL in the SABPSAMP member ABPWKED.

DSNUTILB intercept process flow

Review this high-level process flow to learn how the DSNUTILB intercept works with DSNUTILB to implement the enhancements or intercept processing feature for DB2 utilities. This information can help you interpret the DSNUTILB intercept output.

The following figure shows the general DSNUTILB process flow. This process flow assumes that you previously created a DSNUTILB intercept policy and started the DB2 UET started task. During started task initialization, the policy is loaded into memory and activated so that it is available for DB2 utility processing that DSNUTILB initiates.

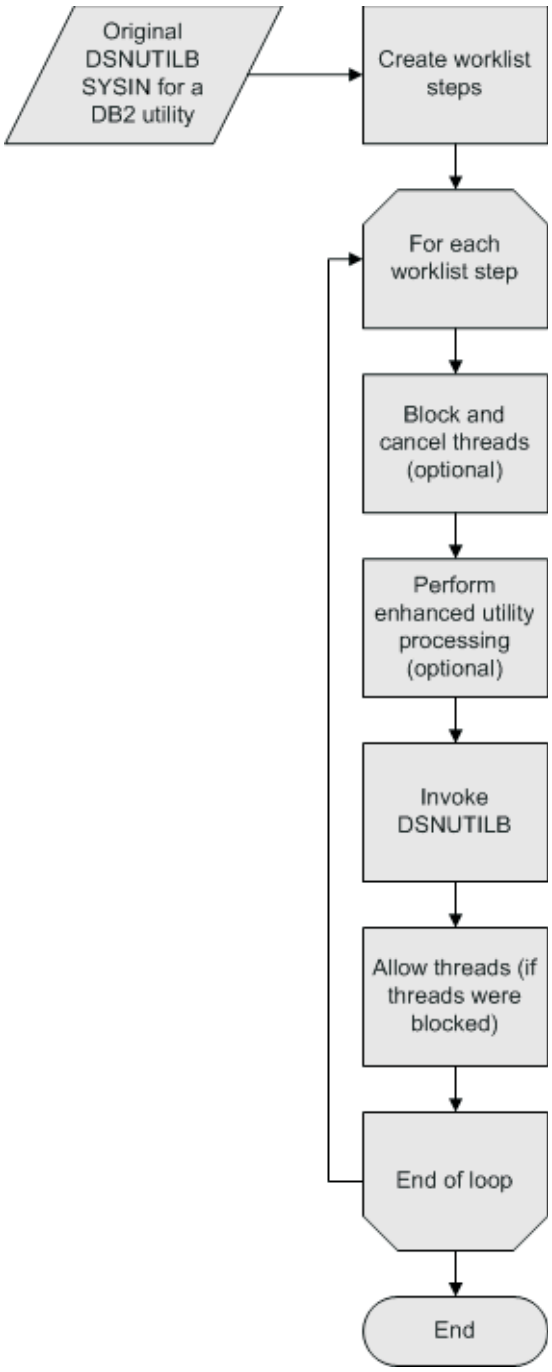


Figure 20. DSNUTILB intercept process flow

Process flow:

1. On each DB2 subsystem that you specified in the policy, the DSNUTILB intercept component intercepts the DSNUTILB program and analyzes the DSNUTILB SYSIN stream for a DB2 utility.
2. The DSNUTILB intercept divides the SYSIN syntax into separate worklist steps. Each step is composed of a single utility command (for example, LOAD) and any applicable setup statements (for example, LISTDEF). All worklist steps that are generated from a single SYSIN stream are associated with the same utility ID. The worklist data is stored across several DB2 tables.

3. For each worklist step (utility command), DB2 UET performs the following processing:
   a. DB2 UET checks the DSNUTILB intercept policy.
   b. If the policy specifies rules that specify DB2 utilities, users, or objects for which to block and cancel threads, DB2 UET cancels active threads and blocks new threads on the appropriate objects. If the original SYSIN stream specified a LISTDEF that identified the DB2 objects for utility processing, these objects must also be selected based on the intercept policy rules for intercept processing to occur for them.
   c. If you configured utility-specific enhancements (that is, the additional options for the LOAD utility or the automatic sizing and creation of the mapping table and mapping-table index for the REORG TABLESPACE utility), DB2 UET runs the utility command and transparently incorporates these enhancements into utility processing.
   d. After the utility command completes, DB2 UET allows new threads to form on the DB2 objects once again (if threads were blocked) and drops any mapping table and mapping-table index that were dynamically created for a REORG TABLESPACE utility.
4. DB2 UET repeats step 3 for each utility command (worklist step).

How the DSNUTILB intercept affects the restart of DB2 utilities

The DSNUTILB intercept supports the normal restart capabilities of all DB2 versions that DB2 UET supports. When a DB2 utility for which interception is occurring terminates abnormally and then you restart it, DB2 resumes utility processing from the appropriate point without any special user intervention.

With a normal DB2 restart, the intercept processing for a utility resumes from the beginning of the worklist step that was being processed when the utility failed. The DSNUTILB intercept does not need to repeat previous worklist steps that were already performed for the failed utility. If you made changes to the utility job to correct a problem that was related to the abnormal termination, the DSNUTILB intercept updates the worklist data for that utility (utility ID) with your changes.

In some exceptional circumstances, you might need to use the DB2 UET ABPMAINT utility to indicate the point at which intercept processing should resume for the DB2 utility when you restart it. This situation can arise when a utility-command operation within a worklist step completes and then an event occurs that causes the utility to end before the completion status of that utility-command operation is recorded in the DB2 UET worklist tables. For example, the event could be an abend of the DB2 UET started task or of DB2, or the issuance of the z/OS Cancel command against the DB2 utility by a user. In such a situation, a normal DB2 restart might not be possible. However, you can use the ABPMAINT utility to control the point at which intercept processing resumes. The ABPMAINT utility provides functions that set the status of the current worklist step or the current utility-command operation within the worklist step to complete so that the DB2 utility restarts either from the beginning of the next worklist step or from the beginning of the next operation in the current worklist step (the next operation after the utility-command operation that was not recorded as complete). The JCL for running the ABPMAINT utility is provided in the *abpid*MNT member that the Tools Customizer created in the *hlq.mlq*.SABPSAMP library. For more information, see “Restarting a DB2 utility in exceptional circumstances” on page 319,

Tip: You can also use the ABPMAINT utility to intentionally terminate a DB2 utility for which interception has occurred and to automatically remove all data that is associated with the utility ID from the worklist tables. See "Terminating a DB2 utility for which interception has occurred" on page 318.

Related tasks:

"Restarting a DB2 utility in exceptional circumstances" on page 319
When a DB2 utility for which DSNUTILB interception is occurring terminates abnormally, DB2 can usually resume utility processing from the appropriate point, without any special user intervention, when you restart the utility. However, in some exceptional circumstances, a normal DB2 restart of a DB2 utility might not be possible. In these circumstances, you can use the DB2 UET ABPMAINT utility to resume utility processing.

Defining and using a DSNUTILB intercept policy

If you want to use the DSNUTILB intercept component to implement the DB2 UET enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads automatically for certain DB2 utility operations, you must define a DSNUTILB intercept policy.

The policy is written in XML, a document markup language that tags elements of a document that then allows a parser to read and understand the instructions in the document. The policy indicates the DB2 subsystems on which to perform intercept processing and any DB2 objects or utilities for which to block and cancel threads or to monitor. You can define only one policy for a DB2 UET started task. This policy addresses all DSNUTILB intercept processing that the started task performs for the DB2 utilities.

DB2 UET provides a sample DSNUTILB policy member (*abpid*PLCY, where *abpid* is the configuration ID that you specified in Tools Customizer) for your use in the *hlq.mlq*.SABPSAMP library. You can customize this member to create a policy that is tailored for your DSNUTILB intercept processing needs. Alternatively, you can create a policy as a sequential file.

For your policy to be used, you must specify the name of the policy member or file in the ABPPLCY DD statement of your started task PROC (*abpid*PROC). Interception will then be enabled by default on the z/OS system on which the DB2 UET started task is running. You can use the z/OS console commands that DB2 UET provides to display the policy and to deactivate or reactivate interception based on the policy. For more information, see "Managing DSNUTILB interception" on page 309.

Related concepts:

"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

"Managing DSNUTILB interception" on page 309
You can manage DSNUTILB interception by performing some routine and occasional tasks.

Related tasks:

"Implementing the LOAD utility enhancements" on page 268
Perform the following steps to implement any of the additional options that DB2 UET supplies for the LOAD utility.

"Implementing the REORG TABLESPACE utility enhancements" on page 292
Perform these steps to have DB2 UET automatically create and drop mapping tables and mapping-table indexes for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option.

Overall structure of the DSNUTILB intercept policy

A DSNUTILB intercept policy must contain one (and only one) <POLICY> section to perform any type of DSNUTILB intercept processing. Optionally, the policy can also contain zero or more <RULESET> sections if you are performing intercept processing.

Example policy

The following example policy is composed of one <RULESET> section that contains exclusion rules and inclusion rules and one <POLICY> section that identifies one DB2 subsystem.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE OPTIONS SYSTEM "DD:DTD(ABPDTDPL)">
<DSNUTILB_INTERCEPT>
<!-- ********************************************************* -->
<!-- * DBP1 RULES                                              * -->
<!-- ********************************************************* -->
<RULESET NAME="DBP1_RULES">
  <EXCLUDE>
      <RULE UTILITY_USERID="INT%"/>
  </EXCLUDE>
  <INCLUDE>
      <RULE UTILITY_USERID="PDADM%"/>
      <RULE UTILITY_USERID="PDMG%"/>
      <RULE TABLESPACE="DB1%.TS1%" PART="1,3,5-10,20-22,17-18"/>
      <RULE DATABASE="DB1%"/>
      <RULE TABLE="ABP%.%"/>
   </INCLUDE>
   <EXCLUDE>
       <RULE INDEXSPACE="DB1%.IND%"/>
   </EXCLUDE>
 </RULESET>
<!-- ********************************************************* -->
<!-- * POLICY                                                  * -->
<!-- ********************************************************* -->
<POLICY>
  <DB2SYSTEM SSID="DBP1">
     <USE_RULESET NAME="DBP1_RULES"/>
     <EXCLUDE>
       <RULE UTILITY_USERID="PDEU%"/>
     </EXCLUDE>
  </DB2SYSTEM>
  </POLICY>
  </DSNUTILB_INTERCEPT>
```

The <RULESET> section defines two sets of exclusion rules and one set of inclusion rules. Only the threads that meet all of the following criteria will be eligible for cancelation: 1) threads that originated from utilities that have any SHRLEVEL value (ALL is assumed when the SHRLEVEL attribute is not specified); 2) threads that are running under user IDs that match the wildcard patterns "PDADM%" or "PDMG%" but *not* "INT%"; 3) threads that are referencing tables that have creator IDs beginning "ABP" and that are in the specified table space partitions and database; and 4) threads that do *not* reference indexes in an index space that has a name beginning "IND" and is in a database with a name beginning "DB1".

The <POLICY> section identifies the DB2 subsystem “DBP1” and assigns the ruleset “DBP1_RULES” to this subsystem. The referenced ruleset is defined in a <RULESET> section of the same policy. Additionally, an <EXCLUDE> rule is specified for excluding utility operations that are run under user IDs that begin with “PDEU”.

For more information about the XML elements that you can use in a policy, see “Reading and understanding the policy.”

Reading and understanding the policy

Within an XML document such as the DB2 UET policy, there are elements, attributes, values, and hierarchical relationships between elements.

Definitions of each of the XML documents used within the DB2 UET policy are defined for comprehensiveness.

Element
A tag, along with its attributes and the text and elements that it encloses, that defines a characteristic of a document. Elements within another element are called *child elements*.

Used by element
Any higher-level element or elements that can contain the subject element that is being defined.

Child element
Tags that define characteristics of a document, but are only specified when the parent element is defined. That is, they cannot be specified on their own, without the parent element being declared first.

Attribute
The name-value pair that immediately follows an element name. Any value that you specify for an attribute must be enclosed in double quotation marks or single quotation marks. Attributes enhance the specification of an element.

In some attribute values, you can include the following DB2 wildcard characters. If you do so, all items that match the wildcard pattern will be selected.

- A percent sign (%) for zero or more characters
- An underscore (_) for a single character

Important: When specifying elements and attributes in your intercept policy, make sure that you do not split the XML characters **</** or **/>** across lines. These characters indicate the end of an element, for example, **</**RULESET> or <RULE DATABASE="*name*"**/>**. If you have DB2 UDB for z/OS Version 8 or DB2 Version 9.1 for z/OS, you can specify values that are longer than the 80-column line length for some element attributes. In this case, type the value up to column 80 and then continue it in column 1 of the next line, without splitting the ending /> characters.

Value An attribute has values or valid options that may be used within an attribute, much like a parameter has options.

<DB2SYSTEM>:

A <DB2SYSTEM> element identifies a DB2 subsystem for which to perform DSNUTILB intercept processing to block and cancel threads or to monitor DB2 utilities. If you specify a generic wildcard pattern as its attribute value, this

element can identify multiple DB2 subsystems. You must specify one or more <DB2SYSTEM> elements within the <POLICY> section. You cannot specify a <DBSYSTEM> element in a <RULESET> section.

Contained by

<POLICY>

Contains

The following elements identify the DB2 utility operations and objects for which to perform thread-cancelation processing on the specified subsystem or subsystems. All elements are optional. You can specify zero or more of these elements.

- <USE_PRACTICE>
- <USE_RULESET>
- <INCLUDE>
- <EXCLUDE>

Attributes

The following table describes the attributes that you can specify for the <DB2SYSTEM> element:

Table 11. <DB2SYSTEM> attributes

Attribute		Default value	Required?
SSID	Indicates a valid subsystem identifier for a DB2 subsystem on which you want to perform a DSNUTILB intercept action. This value can be up to four characters long. No default value is provided. Wildcards are permitted. **Tip:** Ensure that the DB2 UET plan is bound on the subsystem that you specify.	None	Yes
ACTION	Indicates the DSNUTILB intercept action that you want to perform for the defined subsystem when evaluating the policy rules. Values: • BLOCK_AND_CANCEL_THREADS • MONITOR_UTILITY • VRUPDATE • AUTO_DIRECTOR For more information, see the following section.	BLOCK_AND_ CANCEL_ THREADS	No

ACTION attribute values

The ACTION attribute assigns a specific action to be taken by DB2 UET for the DB2 system that is defined by the <DB2SYSTEM> element. You can specify the same ACTION value only once per DB2 system. The following ACTION values are valid.

BLOCK_AND_CANCEL_THREADS specifies that threads should be blocked and canceled based on the criterion defined in the elements RULE and RULESET. DB2 UET also blocks additional threads from coming active on those objects to allow a utility to run. When a utility has completed, DB2 UET will resume allowing threads to access the objects. BLOCK_AND_CANCEL_THREADS can only be used with policy rules. If you want to specify VRUPDATE for the same DB2SSID as BLOCK_AND_CANCEL_THREADS, then the DB2SYSTEM element must be specified twice in the policy, once for each action. For more information, see the examples that follow.

MONITOR_UTILITY specifies the utility or utilities on which syntax should be monitored, and is invoked when the criterion in the RULE and RULESET matches the utility being executed. For more information, see Chapter 6, “Monitoring DB2 utility statements for text strings, events, and messages,” on page 245.

VRUPDATE instructs DB2 UET to customize DB2 Object Restore JCL any time a LOAD or REORG TABLESPACE utility with the LOG YES option successfully completes. The JCL data set that is specified in the DSN attribute of the VRUPDATE element is customized and then passed to a JES internal reader. If the DSN attribute of the VRUPDATE element is omitted, then the parser will issue a warning message in the started task SYSPRINT log.

The subparameter SUBMIT_FROM_SERVER specifies whether to submit the VRUPDATE job under the authority of the DB2 UET started task, or under the authority of the USERID of the submitter of the utility job.

Within the policy rules:

- If SUBMIT_FROM_SERVER is set to the default, No, then the VRUPDATE job is submitted under the authority of the USERID that submitted the utility job.
- If SUBMIT_FROM_SERVER is set to Yes, then the VRUPDATE job is submitted under the authority of the DB2 UET started task.
- The absence of this subparameter is equivalent to specifying SUBMIT_FROM_SERVER=NO within the policy.

Because you can customize the VRUPDATE job, and the started task policy includes the ACTION and DSN parameters, run either a LOAD utility with the option LOG YES specified, or run a REORG utility with LOG YES specified.

The DB2 UET started task will intercept the DSNUTILB program and parse through the utility syntax. When LOG YES is detected either on the LOAD utility or the REORG utility, a VRUPDATE job will be customized using the member specified in the DSN parameter in the policy; the original VRUPDATE job in the DSN data set will remain untouched. One VRUPDATE job will be customized and submitted to the internal reader per utility. Within the VRUPDATE job, the character string ‘????' will resolve to the DB2 subsystem on which the LOAD or REORG utility ran, and the character string ‘####' will resolve to the table space name defined in the LOAD or REORG utility.

(Available with the DB2 Utilities Solution Pack only.) AUTO_DIRECTOR enables collection of utility execution data. Specify this ACTION value for each DB2 system that is to be intercepted for the collection of utility execution data.

Examples

The following example identifies all subsystems that have SSIDs beginning with "DB2." On these subsystems, the "DEFAULT_RULES" ruleset will be used. This ruleset contains the default rules for selecting the DB2 objects for which to cancel threads.

```
<DB2SYSTEM SSID="DB2%">
  <USE_RULESET NAME="DEFAULT_RULES"/>
</DB2SYSTEM>
```

The following example specifies the action BLOCK_AND_CANCEL_THREADS in combination with a policy rule.

```
<DB2SYSTEM SSID="DB2A"
ACTION="BLOCK_AND_CANCEL_THREADS">
<INCLUDE>
   <RULE UTILITY_USERID="TSOUSR%"/>
</INCLUDE>
</DB2SYSTEM>
```

The following example specifies the actions BLOCK_AND_CANCEL_THREADS and VRUPDATE for the same DB2SSID. The DB2SYSTEM element must be specified twice in the policy, once for each action.

```
<DB2SYSTEM SSID="DB2A"
ACTION="BLOCK_AND_CANCEL_THREADS">
<INCLUDE>
   <RULE UTILITY_USERID="TSOUSR%"/>
</INCLUDE>
</DB2SYSTEM>
<DB2SYSTEM SSID="DB2A" ACTION="VRUPDATE">
   <VRUPDATE DSN="ABP.VRUPDATE.JCL"/>
</DB2SYSTEM>
```

The following examples shows the use of SUBMIT_FROM_SERVER:

```
<DB2SYSTEM SSID="D8B" ACTION="VRUPDATE">
<VRUPDATE DSN="ABP.VRUPDATE.JCL"
SUBMIT_FROM_SERVER="YES"/>
</DB2SYSTEM>
```

The following example specifies the action VRUPDATE and the VRUPDATE JCL.

```
<DB2SYSTEM SSID="DB2A" ACTION="VRUPDATE">
<VRUPDATE DSN="ABP.V210.SABPSAMP"
</DB2SYSTEM>
```

The resultant diagnostic output follows:

```
//VRUPDATE EXEC PGM=AUO#UTIL,PARM='VRUPDATE,????'
//*
//STEPLIB DD DISP=SHR,DSN=<installation_must_provide>
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
//DB2PARMS DD DISP=SHR,DSN=<installation_must_provide>
//*
//BKUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) PACKAGES REPORT
//BPUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) PLANS REPORT
//CAUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) COLUMN AUTH REPORT
//DAUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) DBAUTH REPORT
//DBUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) DATABASE REPORT
//DTUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) DATATYPES REPORT
```

```
//IXUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) INDEX REPORT
//PAUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) PKG AUTH REPORT
//PLUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) PLAN AUTH REPORT
//PMUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) DBRM REPORT
//RAUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) USER AUTH REPORT
//SCUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) IMAGE COPY REPORT
//SGUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) STOGROUP REPORT
//SPUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) RES AUTH REPORT
//SRUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) ROUTINES REPORT
//SYUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) SYNONYM REPORT
//TAUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) TABLE AUTH REPORT
//TBUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) TABLE REPORT
//TRUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) TRIGGERS REPORT
//TSUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) TABLESPACE REPORT
//UPUL#RPT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133) PROCEDURES REPORT
//*
//SYSIN DD *
UPDATE=ON
SYSCOPYONLY=ON
TABLESPACE=####
/*
//
```

(Available with the DB2 Utilities Solution Pack only.) The following example specifies the action AUTO_DIRECTOR in combination with the RULE element.

```
<DB2SYSTEM SSID="SSID" ACTION="AUTO_DIRECTOR">
  <INCLUDE>
     <RULE UTILITY_USERID="USER%"/>
  </INCLUDE>
</DB2SYSTEM>
```

The RULE element provides a filter for the collection of utility execution data. In this example, <RULE UTILITY_USERID="USER%"> filters the collection to only utilities that were run by any user ID with USER as the first four characters. <RULE UTILITY_USERID="%"> enables collection for all utilities executed by any user ID.

Related concepts:

"Example policy to invoke the Utility Monitor" on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

Related reference:

"<USE_PRACTICE>" on page 154
A <USE_PRACTICE> element specifies the name of the predefined practice to be used when evaluating the policy rules within the element <DB2SYSTEM>, and can only be specified once.

"<USE_RULESET>" on page 155
A <USE_RULESET> element assigns a ruleset that you defined for thread-cancelation processing to a DB2 subsystem.

"<INCLUDE> and <EXCLUDE>" on page 136
An <INCLUDE> element contains one or more rules for including DB2 objects or utility operations in thread-cancelation processing. An <EXCLUDE> element contains one or more rules for excluding DB2 utilities, users, or objects from thread-cancelation processing.

"<POLICY>" on page 141
The <POLICY> element is required for all types of processing that you can perform by using the DSNUTILB intercept. The <POLICY> element identifies the DB2 subsystems on which to implement the enhancements for the LOAD and REORG TABLESPACE utility or intercept processing.

<DSNUTILB_INTERCEPT>:

The <DSNUTILB_INTERCEPT> element is the top-level container for all other elements that comprise a DSNUTILB intercept policy. You must specify the <DSNUTILB_INTERCEPT> element at the beginning of an intercept policy that you define. Place the starting tag immediately after the XML and DOCTYPE declarations, and place the ending tag after all other elements in the policy. In XML terms, this is called the *root element*.

Used by elements

No other element. <DSNUTILB_INTERCEPT> is the root element, or the top-level element for a policy.

Child elements

One <POLICY> element is required. You can specify zero or more of the following optional elements:

- <RULESET>
- <PRACTICE>
- <POLICY>

Attributes

None.

Related reference:

"<POLICY>" on page 141
The <POLICY> element is required for all types of processing that you can perform by using the DSNUTILB intercept. The <POLICY> element identifies the DB2 subsystems on which to implement the enhancements for the LOAD and REORG TABLESPACE utility or intercept processing.

"<RULESET>" on page 150
A <RULESET> element defines a set of rules for determining the DB2 utilities, users, and objects for which to block and cancel threads, and utilities that are monitored as well as what syntax to enforce.

"<PRACTICE>" on page 142
A <PRACTICE> element enables the DSNUTILB intercept feature of DB2 UET to search the provided DB2 utility statements and perform specified actions.

<INCLUDE> and <EXCLUDE>:

An <INCLUDE> element contains one or more rules for including DB2 objects or utility operations in thread-cancelation processing. An <EXCLUDE> element contains one or more rules for excluding DB2 utilities, users, or objects from thread-cancelation processing.

You can optionally specify zero or more <INCLUDE> or <EXCLUDE> elements in a <RULESET> section, the <POLICY> section, or both.

One of these elements is required both before and after a <RULE> or <RULESET> element. If you specify the <INCLUDE> or <EXCLUDE> element under a <RULESET> element, you must assign the ruleset to one or more DB2 subsystems in the <POLICY> section by specifying the <USE_RULESET> element.

In the <POLICY> section, you must specify the <INCLUDE> or <EXCLUDE> element under a <DB2SYSTEM> element that specifies a valid DB2 SSID. You can define additional <INCLUDE> or <EXCLUDE> elements under a <DB2SYSTEM> element to refine thread-selection criteria for a DB2 subsystem.

Used by elements

- <RULESET>
- <DB2SYSTEM>

Child elements

At least one <RULE> element is required. You can specify one or more of these elements. If you specify an <EXCLUDE> or <INCLUDE> element, you must specify at least one <RULE> element under it. You can specify multiple <RULE> elements if appropriate.

Attributes

None.

<INCLUDE> example

The following example selects RECOVER utility operations on the DB2 subsystem "DB123" for thread-cancelation processing when the RECOVER command is issued under the user ID of either "DBA1" or "DBA2."

```
<POLICY>
  <DB2SYSTEM SSID="DB123">
    <INCLUDE>
      <RULE UTILITY_USERID="DBA1"/>
      <RULE UTILTIY_USERID="DBA2"/>
      <RULE UTILITY_COMMAND="RECOVER"/>
    </INCLUDE>
  </DB2SYSTEM>
</POLICY>
```

<EXCLUDE> example

The following example defines a group of exclusion rules for a ruleset. These rules exclude the "TSABC" table space from thread-cancelation processing when a configuration of that table space occurs in either the "DB123" or "DB456" database.

```
<RULESET Name="MYEXCLUDE_RULES">
  <EXCLUDE>
    <RULE DATABASE="DB123"/>
    <RULE DATABASE="DB456"/>
    <RULE TABLESPACE="TSABC"/>
  </EXCLUDE>
</RULESET>
```

Related reference:

"<DB2SYSTEM>" on page 131
A <DB2SYSTEM> element identifies a DB2 subsystem for which to perform DSNUTILB intercept processing to block and cancel threads or to monitor DB2 utilities. If you specify a generic wildcard pattern as its attribute value, this element can identify multiple DB2 subsystems. You must specify one or more <DB2SYSTEM> elements within the <POLICY> section. You cannot specify a <DBSYSTEM> element in a <RULESET> section.

"<RULESET>" on page 150
A <RULESET> element defines a set of rules for determining the DB2 utilities, users, and objects for which to block and cancel threads, and utilities that are monitored as well as what syntax to enforce.
"<RULE>" on page 144
A <RULE> element provides a criterion for determining the DB2 utilities, users, or object types for which to perform intercept processing.

<MESSAGE>:

A <MESSAGE> element defines and specifies messages to the Message Monitor.

Used by elements

For the specified DB2 message, DB2 UET can change the return code, suppress it from being written to the SYSPRINT DD, or journal the action to the DB2 UET tables. Messages can be specified for single utility commands or for all utilities. Messages are parsed by the DSNUTILB intercept at utility run-time as they are written to utility job SYSPRINT.

<MONITOR>

Child elements

No other elements.

Attributes

You can specify the following attributes for the <MESSAGE> element.

Table 12. <MESSAGE> attributes

Attribute	Default value	Required?
ID Indicates the message ID, for example, DSNU350I. Wildcards are *not* permitted.	None	Yes
RETURN_CODE Changes the final return code passed back from DB2 UET to DSNUTILB after the DB2 UET intercept execution of the utility statement. Wildcards are *not* permitted.	None	No
JOURNAL Controls whether the message text is recorded in the Utility Monitor journal table. Valid values: YES, NO If you specify spaces for the value or an empty value, then DB2 UET uses the default value.	YES	No

Table 12. <MESSAGE> attributes (continued)

Attribute	Default value	Required?
SUPPRESS Specifies whether the message is suppressed from the output. Valid values: YES, NO If you specify spaces for the value or an empty value, then DB2 UET uses the default value.	NO	No
WSL Controls whether Message Monitor actions are applied to utility programs that are run under the control of DB2 Administration Tool work statement list (WSL) processing. Valid values: YES, NO If you specify spaces for the value or an empty value, then DB2 UET uses the default value.	NO	No
JOURNAL_LIMIT Specifies the number of duplicate message IDs that are logged in the Utility Monitor journal tables by batch jobs. The value must be a non-negative integer in the range of 0 through 2147483647. The Message Monitor interprets the value of 0 and any value outside the valid range as no limit. For the Message Monitor to use the JOURNAL_LIMIT attribute, the value of the JOURNAL attribute must be YES.	None	No

Example

```
<MONITOR>
  <SYNTAX VALUE="LOG YES" SUBSTITUTE="LOG NO"/>
  <MESSAGE ID="DSNU350I" RETURN_CODE="12" JOURNAL="NO"/>
</MONITOR>
```

Related reference:

"<MONITOR>"
A element <MONITOR> identifies a set of syntax rules by which to evaluate the specified DB2 utility. This element is required when invoking the Utility Monitor and can only be specified once per <UTILITY> element.

<MONITOR>:

A element <MONITOR> identifies a set of syntax rules by which to evaluate the specified DB2 utility. This element is required when invoking the Utility Monitor and can only be specified once per <UTILITY> element.

Used by elements

<UTILITY>

Child elements

<SYNTAX>

<MESSAGE>

Attributes

You can specify the following attributes for the <MONITOR> element.

Table 13. <MONITOR> attributes

Attribute	Default value	Required?
JOURNAL Specifies that logging is to take place to track the actions of the Utility Monitor activities. This attribute is only specified once with either element <MONITOR> or <SYNTAX>. If JOURNAL is specified with the <MONITOR> element, it will take precedence for all <SYNTAX> elements contained within the <MONITOR> element. If JOURNAL is specified on the <SYNTAX> element, it will override any value specified in the <MONITOR> element for that syntax activity only. Values: • YES: A row is written to theDB2 UET Journal tables. • NO: No journaling activity will occur.	YES	No

Table 13. <MONITOR> attributes (continued)

Attribute		Default value	Required?
FAIL	Specifies whether to fail the job step when passing control to DSNUTILB. This attribute is only specified once with either element <MONITOR> or <SYNTAX>. Valid values are an integer in the range of 8 to 4095. For example, FAIL="20". The utility job step return code will be set to the value defined by the attribute FAIL. If FAIL is specified for the <MONITOR> element, the value becomes the default value for all subsequent <SYNTAX> elements but can be overridden if it is also specified in the child element <SYNTAX>.	None	No

Example

```
<UTILITY NAME="REORG_TABLESPACE">
         <MONITOR>
           <SYNTAX REMOVE="UNLOAD PAUSE"/>
           <SYNTAX VALUE="SCOPE %" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"/>
           <SYNTAX VALUE="LOG NO" SUBSTITUTE="LOG YES"/>
           <SYNTAX VALUE="SHRLEVEL %" SUBSTITUTE="SHRLEVEL NONE"/>
           <SYNTAX VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 12"/></p><p>
           <SYNTAX ADD="KEEPDICTIONARY"/>
           <MESSAGE ID="DSNU350I" RETURN_CODE="12" JOURNAL="NO"/>
         </MONITOR>
</UTILITY>
```

Related concepts:

"Example policy to invoke the Utility Monitor" on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

Related reference:

"<SYNTAX>" on page 151
The element <SYNTAX> specifies the syntax to monitor.

"<MESSAGE>" on page 138
A <MESSAGE> element defines and specifies messages to the Message Monitor.

"<UTILITY>" on page 156
The element <UTILITY>, when used in conjunction with the attribute NAME, identifies the DB2 utility that DB2 UET is to evaluate when DSNUTILB is invoked to verify whether the Practice rule should be enforced.

<POLICY>:

The <POLICY> element is required for all types of processing that you can perform by using the DSNUTILB intercept. The <POLICY> element identifies the DB2 subsystems on which to implement the enhancements for the LOAD and REORG TABLESPACE utility or intercept processing.

If you do not specify at least one DB2 subsystem, no intercept processing will occur. Optionally, the <POLICY> element also associates the DB2 subsystems with

any rules and rulesets that you define for selecting DB2 utilities, users, or objects for thread-cancelation processing or utility monitoring. You can specify only one <POLICY> element within a DSNUTILB intercept policy.

Used by elements

<DSNUTILB_INTERCEPT>

Child elements

At least one <DB2SYSTEM> element is required. If you include a wildcard character in the attribute value for a <DB2SYSTEM> element, all subsystems that match the wildcard pattern will be selected.

Under a <DB2SYSTEM> element, you can optionally specify any number of <USE_RULESET>, <EXCLUDE>, and <INCLUDE> elements to identify the DB2 utilities, users, or objects for which to block and cancel threads or perform utility monitoring. The <USE_RULESET> element references a named ruleset (group of rules) that is defined under a <RULESET> element of the same intercept policy. The <EXCLUDE> and <INCLUDE> elements contain specific exclusion or inclusion rules that you want to use for the specified subsystem.

Attributes

None.

Related reference:

"<DB2SYSTEM>" on page 131
A <DB2SYSTEM> element identifies a DB2 subsystem for which to perform DSNUTILB intercept processing to block and cancel threads or to monitor DB2 utilities. If you specify a generic wildcard pattern as its attribute value, this element can identify multiple DB2 subsystems. You must specify one or more <DB2SYSTEM> elements within the <POLICY> section. You cannot specify a <DBSYSTEM> element in a <RULESET> section.

"<DSNUTILB_INTERCEPT>" on page 136
The <DSNUTILB_INTERCEPT> element is the top-level container for all other elements that comprise a DSNUTILB intercept policy. You must specify the <DSNUTILB_INTERCEPT> element at the beginning of an intercept policy that you define. Place the starting tag immediately after the XML and DOCTYPE declarations, and place the ending tag after all other elements in the policy. In XML terms, this is called the *root element*.

"<UTILITY>" on page 156
The element <UTILITY>, when used in conjunction with the attribute NAME, identifies the DB2 utility that DB2 UET is to evaluate when DSNUTILB is invoked to verify whether the Practice rule should be enforced.

<PRACTICE>:

A <PRACTICE> element enables the DSNUTILB intercept feature of DB2 UET to search the provided DB2 utility statements and perform specified actions.

The element <PRACTICE> identifies a set of specifications that allow utility syntax to be changed at runtime by the Utility Monitor. You can define this element multiple times within the element <DSNUTILB_INTERCEPT>. This element can be defined only once within the element <DB2SYSTEM>.

For more information, see Chapter 6, “Monitoring DB2 utility statements for text strings, events, and messages,” on page 245.

Used by elements

<DSNUTILB_INTERCEPT>

Child elements

<UTILITY>

Attributes

You can specify the following attribute for the <PRACTICE> element.

Table 14. <PRACTICE> attributes

Attribute		Default value	Required?
NAME	Specifies the name of the RULE that contains subsequent syntax actions. This element is required when invoking the Utility Monitor. The maximum length of the name is 32 characters.	None	Yes

Example

```
<PRACTICE NAME="CHECK DATA">
   <UTILITY NAME="CHECK DATA">
     <MONITOR>
       <SYNTAX ADD="SORTNUM 10" OPTIONIF="SORTDEVT"/>
     </MONITOR>
   </UTILITY>
</PRACTICE>
```

For more information, see Chapter 6, “Monitoring DB2 utility statements for text strings, events, and messages,” on page 245.

Related concepts:

“Example policy to invoke the Utility Monitor” on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

Related reference:

“<DSNUTILB_INTERCEPT>” on page 136
The <DSNUTILB_INTERCEPT> element is the top-level container for all other elements that comprise a DSNUTILB intercept policy. You must specify the <DSNUTILB_INTERCEPT> element at the beginning of an intercept policy that you define. Place the starting tag immediately after the XML and DOCTYPE declarations, and place the ending tag after all other elements in the policy. In XML terms, this is called the *root element*.

“<UTILITY>” on page 156
The element <UTILITY>, when used in conjunction with the attribute NAME, identifies the DB2 utility that DB2 UET is to evaluate when DSNUTILB is invoked to verify whether the Practice rule should be enforced.

<RULE>:

A <RULE> element provides a criterion for determining the DB2 utilities, users, or object types for which to perform intercept processing.

You must define at least one <RULE> under an <INCLUDE> or <EXCLUDE> element in a <RULESET> section or the <POLICY> section of the DSNUTILB intercept policy before intercept processing can occur. This element may be defined multiple times within the element DSNUTILB_INTERCEPT.

The <RULE> elements do not apply to the utility-specific enhancements that are implemented by means of the DSNUTILB intercept, such as the LOAD and REORG TABLESPACE enhancements.

The string value is the name of the object type specified in the RULE. For valid object type attributes, see Table 17 on page 146.

Used by elements

- <INCLUDE>
- <EXCLUDE>

Child elements

No other elements.

Attributes

You can use the standard DB2 wildcard characters in some attribute values to select all items that match a wildcard pattern.

If you use DB2 UDB for z/OS Version 8 or later or DB2 Version 9.1 or later for z/OS, you can specify values that are longer than the 80-column line length for some attributes. To do so, type the value up to column 80 and then continue the value in column 1 of the next line. However, do not split the closing XML characters /> at the end of the RULE definition.

The following table describes the attributes that you can specify for the <RULE> element. The **Description and values** column indicates whether wildcards are supported.

Utility environment RULE elements are matched with runtime environment variables such as JOB or EXEC statement parameters. These RULE elements relate to all utility statements. The following table describes the Utility environment attributes that you can specify for the <RULE> element.

Table 15. Utility environment <RULE> attributes

Attribute	Default value	Required?
UTILITY_ID The DB2 UTILID identifier for the DB2 utility that you want to include in or exclude from intercept processing. This ID can be up to 16 characters long. Wildcards are permitted.	None	No

Table 15. Utility environment <RULE> attributes (continued)

Attribute	Default value	Required?
UTILITY_JOBNAME The job name for the DB2 utility that you want to include in or exclude from intercept processing. This name can be up to eight characters long. Wildcards are permitted.	None	No
UTILITY_USERID The user ID of the person who is running the DB2 utility that you want to include in or exclude from intercept processing. This ID can be up to eight characters long. Wildcards are permitted.	None	No

Utility syntax <RULE> elements are matched with data found in the utility control statements. The following table describes the Utility syntax attributes that you can specify for the <RULE> element.

Table 16. Utility syntax <RULE> attributes

Attribute	Default value	Required?
SHRLEVEL The SHRLEVEL option that is specified in the utility statement that you want to include in or exclude from intercept processing. The SHRLEVEL option indicates the type of concurrent access that other applications have to a DB2 object while a utility is running. Valid values are: • ALL: Utility statements that specify any SHRLEVEL option. • CHANGE: Only utility statements that specify SHRLEVEL CHANGE (which allows concurrent read and write operations). • NONE: Only utility statements that specify no SHRLEVEL option. • REFERENCE: Only utility statements that specify SHRLEVEL REFERENCE (which allows concurrent read operations but not write operations). Wildcards are *not* permitted.	None	No

Table 16. Utility syntax <RULE> attributes (continued)

Attribute	Default value	Required?
UTILITY_COMMAND The DB2 utility command to include in or exclude from intercept processing. Valid values are: • ALL • CHECK_DATA • CHECK_INDEX • CHECK_LOB • COPY • EXEC_SQL_ALTER • EXEC_SQL_DROP • LOAD • QUIESCE • REBUILD_INDEX • RECOVER • REORG_INDEX • REORG_INDEXSPACE • REORG_TABLESPACE • RUNSTATS • UNLOAD Wildcards are *not* permitted.	None	No

Utility object <RULE> elements are matched with DB2 object names in the utility control statements. The following table lists the Utility object attributes that you can specify for the <RULE> element. These attributes are described in more detail in the section following the table.

Table 17. Utility object <RULE> attributes

Attribute	Default value	Required?
ALIAS A DB2 alias for the tables that you want to include in or exclude from intercept processing. Wildcards are permitted in each part of the value. For more information, see the next section.	None	No
DATABASE The name of the database that contains the DB2 objects that you want to include in or exclude from intercept processing. Wildcards are permitted. For more information, see the next section.	None	No

Table 17. Utility object <RULE> attributes (continued)

Attribute	Default value	Required?
INDEX A DB2 index that you want to include in or exclude from intercept processing. Wildcards are permitted in each part of the value. For more information, see the next section.	None	No
INDEXSPACE The name of an index space that you want to include in or exclude from intercept processing. Wildcards are permitted in each part of the value. For more information, see the next section.	None	No
PART One or more partitions of a table space, an index space, or an index. Wildcards are *not* permitted. For more information, see the next section.	None	No
TABLE A DB2 table that you want to include in or exclude from intercept processing. Wildcards are permitted in each part of the value. For more information, see the next section.	None	No
TABLESPACE The name of a DB2 table space that you want to include in or exclude from intercept processing. Wildcards are permitted in each part of the value. For more information, see the next section.	None	No
VIEW The name of a DB2 view that is associated with the table that you want to include in or exclude from intercept processing. Wildcards are permitted in each part of the value. For more information, see the next section.	None	No

Utility object <RULE> attributes

ALIAS: You can specify an alias name only, or both an alias name and a creator ID in the format *creator_id.alias_name*. An alias name can be up to 18 characters long in DB2 version 7 or up to 128 characters long in later DB2 versions. A creator ID can be up to 128 characters long. Wildcards are permitted in each part of the value. If you omit the creator ID, all aliases that match the specified alias name are selected, regardless of their creator IDs.

DATABASE: A database name can be up to eight characters long. Wildcards are permitted. If you do not specify a DATABASE value, the % wildcard is used by default to indicate all databases.

Important: For thread selection, the DATABASE attribute value that you specify is matched only against two-part table space names and index space names that include a database name (that is, names that have the format *database_name.object_name*). This attribute is ignored for other DB2 objects.

INDEX: You can specify an index name only, or both an index name and a creator ID in the format *creator_id.index_name*. An index name can be up to 128 characters long. A creator ID can be up to 128 characters long. Wildcards are permitted in each part of the value. If you omit the creator ID, all indexes that match the specified index name are selected, regardless of their creator IDs. If you specify an index name in an inclusion rule, DB2 UET will cancel any threads that hold locks on the index space that contains the index, or if no such threads are detected, DB2 UET will cancel threads that hold locks on the table space that contains the table on which the index was created. If you want to specify index partitions, include the PART attribute in the INDEX rule.

Note: For the PART attribute to be valid, you must pair it with the TABLESPACE, INDEXSPACE, or INDEX attributes. Using the PART attribute alone is not valid, will not produce an error or warning message, and will just be ignored.

INDEXSPACE: You can specify an index space name only, or both an index space name and a database name in the format *database_name.index_space_name*. An index space name and a database name can each be up to eight characters long. Wildcards are permitted in each part of the value. Include a database name if you want to specify a particular configuration of the index space. If you want to specify all configurations of the index space in all databases, do not specify a database name, or specify only the % wildcard for the database name in this attribute. Any value that you specify for the DATABASE attribute is processed separately based on the order in which it appears in the policy.

If the original DB2 utility command specifies an INDEXSPACE name without a database name, DB2 will use the DSNDB04 database by default.

Note: For the PART attribute to be valid, you must pair it with the TABLESPACE, INDEXSPACE, or INDEX attributes. Using the PART attribute alone is not valid, will not produce an error or warning message, and will just be ignored. If you want to specify index space partitions, include the PART attribute in the INDEXSPACE rule.

If you specify an index space name in an inclusion rule, DB2 UET will cancel any threads that hold locks on the index space, or if no such threads are detected, DB2 UET will cancel threads that hold locks on the table space that contains the table on which the index was created.

PART: You must specify the PART attribute in the same rule as the TABLESPACE attribute, the INDEXSPACE attribute, or the INDEX attribute. For example:

```
<RULE TABLESPACE="DBP1.TS1"
PART="1,13:24,50:75"/>

<RULE INDEXSPACE="ABP%.ABPIXSP1"
PART="1,3:5,6"/>

<RULE INDEX="ABP%.ABPINDX1"
PART="1:3,6"/>
```

A partition number can be an integer from 1 through 4096 for DB2 Version 8 or later. Specify multiple partitions as follows:

- A series of individual partitions: *integer1*,*integer2*,*integer3*
- A range of partitions: *integer1*:*integer3* (The first value must be less than the second value.)
- Both ranges and individual partitions: *integer1*,*integer2*:*integer4*,*integer5* (The first value must be less than the second value.)

For more information on how PART may be specified within a RULE, refer to "Defining and using a DSNUTILB intercept policy" on page 129.

TABLE: You can specify a table name only, or both a table name and a creator ID in the format *creator_id*.*table_name*. A table name can be up to 18 characters long in DB2 version 7 or up to 128 characters long in later DB2 versions. A creator ID can be up to 128 characters long. Wildcards are permitted in each part of the value. If you omit the creator ID, all tables that match the specified table name are selected, regardless of their creator IDs.

TABLESPACE: You can specify a table space name only or both a table space name and a database name in the format *database_name*.*tablespace_name*. A table space name and a database name can each be up to eight characters long. Wildcards are permitted in each part of the value. Include a database name if you want to specify a particular configuration of the table space. If you want to specify all configurations of the table space in all databases, do not specify a database name or specify only the % wildcard for the database name in this attribute. Any value that you specify for the DATABASE attribute is processed separately based on the order in which it appears in the policy. If the original DB2 utility command specifies a TABLESPACE value without a database name, DB2 will use the DSNDB04 database by default.

Note: For the PART attribute to be valid, you must pair it with the TABLESPACE, INDEXSPACE, or INDEX attributes. Using the PART attribute alone is not valid, will not produce an error or warning message, and will be ignored. If you want to specify table space partitions, include the PART attribute in the TABLESPACE rule.

VIEW: You can specify a view name only, or both a view name and a creator ID in the format *creator_id*.*view_name*. A view name can be up to 18 characters long in DB2 version 7 or up to 128 characters long in later DB2 versions. A creator ID can be up to 128 characters long. Wildcards are permitted in each part of the value. If you omit the creator ID, all views that match the specified view name are selected, regardless of their creator IDs.

Example

The following example cancels threads for the DB2 utility jobs that meet all of the following criteria: 1) they have job names beginning with "JOBABC," 2) they are run under the user ID "DBA1," and 3) they are run against the "PAYROLL" database.

```
<INCLUDE>
  <RULE UTILITY_JOBNAME="JOBABC%"/>
  <RULE UTILITY_USERID="DBA1"/>
  <RULE DATABASE="PAYROLL"/>
</INCLUDE>
```

Related reference:

"<INCLUDE> and <EXCLUDE>" on page 136
An <INCLUDE> element contains one or more rules for including DB2 objects or utility operations in thread-cancelation processing. An <EXCLUDE> element contains one or more rules for excluding DB2 utilities, users, or objects from thread-cancelation processing.

"<RULESET>"
A <RULESET> element defines a set of rules for determining the DB2 utilities, users, and objects for which to block and cancel threads, and utilities that are monitored as well as what syntax to enforce.

<RULESET>:

A <RULESET> element defines a set of rules for determining the DB2 utilities, users, and objects for which to block and cancel threads, and utilities that are monitored as well as what syntax to enforce.

Used by elements

You can define zero or more <RULESET> sections in a DSNUTILB intercept policy. Rulesets are optional and pertain only to thread-cancelation processing and the Utility Monitor. A ruleset is a simply a named group of inclusion and exclusion rules that can be associated with a DB2 subsystem in the <POLICY> section. For a ruleset to be used, you must assign it to one or more DB2 subsystems in the <POLICY> section by using the <USE_RULESET> element. The <RULESET> element is optional, but either <RULE> or <RULESET> must be defined at least once in a policy.

<DSNUTILB_INTERCEPT>

Child elements

At least one of the following elements is required. You can specify one or more of these elements.

- <INCLUDE>
- <EXCLUDE>

You must specify at least one <RULE> element under each <INCLUDE> element and each <EXCLUDE> element. You can specify multiple <RULE> elements if appropriate. Each rule provides an attribute value that is used as selection criterion. Some attribute values can contain wildcards. In this case, all DB2 objects or utility operations that match the wildcard pattern are selected. For more information, see "<RULE>" on page 144.

Attributes

The following table describes the attribute that you can specify for the <RULESET> element:

Table 18. <RULESET> attributes

Attribute	Default value	Required?
NAME Provides a label for the entire ruleset. This name will be referenced from the <POLICY> section. No limit on the name length exists.	None	Yes

Tip: If you define a ruleset name that is longer than the 80-column line length, you can type the name value up to column 80 and then continue it in column 1 of the next line. No special line-continuation characters are required.

Example

The following example defines a ruleset named "INTERN." It contains only one exclusion rule for disallowing the interception of utility jobs that are run by student interns who have user IDs beginning with "INTERN."

```
<RULESET NAME="INTERN">
  <EXCLUDE>
    <RULE UTILITY_USERID="INTERN%"/>
  </EXCLUDE>
</RULESET>
```

Related reference:

"<DSNUTILB_INTERCEPT>" on page 136
The <DSNUTILB_INTERCEPT> element is the top-level container for all other elements that comprise a DSNUTILB intercept policy. You must specify the <DSNUTILB_INTERCEPT> element at the beginning of an intercept policy that you define. Place the starting tag immediately after the XML and DOCTYPE declarations, and place the ending tag after all other elements in the policy. In XML terms, this is called the *root element*.

"<INCLUDE> and <EXCLUDE>" on page 136
An <INCLUDE> element contains one or more rules for including DB2 objects or utility operations in thread-cancelation processing. An <EXCLUDE> element contains one or more rules for excluding DB2 utilities, users, or objects from thread-cancelation processing.

"<RULE>" on page 144
A <RULE> element provides a criterion for determining the DB2 utilities, users, or object types for which to perform intercept processing.

<SYNTAX>:

The element <SYNTAX> specifies the syntax to monitor.

The element <SYNTAX> may be specified multiple times within the element <MONITOR>. If the element <SYNTAX> is specified, at least one of the attributes described in the attributes table is required.

Used by elements

<MONITOR>

Child elements

No other elements.

Attributes

The following table describes the attributes that you can specify for the <SYNTAX> element.

Table 19. <SYNTAX> attributes

Attribute	Default value	Required?

ADD A text string to be added to the utility statement if the specified string is not found in the utility statement syntax. The text string is appended to the end of the utility statement.
 	
This action occurs for each qualifying utility statement. If multiple utility statements were specified, the added text would be appended to each utility statement in the SYSIN. For example:
`//SYSIN DD *`
`REBUILD INDEX ( <IXcreator>.<IXname1> ) ==>added text`
`REBUILD INDEX ( <IXcreator>.<IXname2> ) ==>added text`
`REBUILD INDEX ( <IXcreator>.<IXname3> ) ==>added text`
`/*`
	None	No

REMOVE Specifies the text string to remove from the utility syntax being executed by DSNUTILB if the text string is present. For example, if the COPY utility contains the parameter SHRLEVEL CHANGE, the parameter can be removed, causing the default value of SHRLEVEL REFERENCE to be used instead. If the text string is not present in the utility syntax, no action is taken. This attribute can only be specified once per SYNTAX element. Multiple occurrences of the text string that are found in the SYSIN will be removed. You can specify a wildcard for this value, as shown in the examples.	None	No
VALUE Specifies the text string on which to match within the utility syntax before one of two actions takes place: substitute the target value for another one, or fail the utility. Either SUBSTITUTE or FAIL must be supplied if VALUE is defined. This attribute may only be specified once per SYNTAX element. You can specify a wildcard for this value, as shown in the examples.	None	No

Table 19. <SYNTAX> attributes (continued)

Attribute	Default value	Required?
SUBSTITUTE Specifies the text string to substitute in the utility if there is a match in the string provided for the attribute VALUE. The substituted text is placed in the same location within the utility statement as the target string. This action can occur multiple times in a given SYSIN. For example, if a LOAD utility has the LOG YES parameter specified, DB2 UET will substitute LOG NO in place of LOG YES and pass it to the LOAD utility for processing.	None	No
FAIL Specifies to fail the job step when passing control to DSNUTILB. The utility job step return code will be set to the value defined by the attribute FAIL. This specification of FAIL will override any value specified in the MONITOR element for this syntax activity only. Valid values are any integer from 8 to 4095. For example, FAIL=”20”.	None	No
JOURNAL Indicates to DB2 UET to write this event to the DB2 UET DB2 table for subsequent review or analysis. The information written to the DB2 UET table will include pertinent information about the utility statement and environment, plus the original syntax specified and substituted syntax that was passed to the IBM utility and processed. If JOURNAL is specified with the element MONITOR, it takes precedence for all SYNTAX elements contained within the MONITOR element. If JOURNAL is specified on the element SYNTAX, it overrides the value that is specified in the MONITOR element for that syntax activity only. Values: • YES: A row will be written to the DB2 UET Journal tables. • NO: No journaling activity will occur. This attribute may only be specified once per SYNTAX element.	YES	No
OPTIONIF Value is a text string that must match an identical string in the utility syntax. Specifies a parameter to be added if another parameter is parsed and found. The specified text string must match an identical string in the utility syntax before ADD, REMOVE, or VALUE/SUBSTITUTE actions are performed.	None	No

Examples

```
<UTILITY NAME="REORG_TABLESPACE">
   <MONITOR>
        <SYNTAX REMOVE="UNLOAD PAUSE"/>
        <SYNTAX VALUE="SCOPE %" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"/>
        <SYNTAX VALUE="LOG NO" SUBSTITUTE="LOG YES"/>
        <SYNTAX VALUE="SHRLEVEL %" SUBSTITUTE="SHRLEVEL NONE"/>
        <SYNTAX VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 12"/>
        <SYNTAX ADD="KEEPDICTIONARY"/>
   </MONITOR>
</UTILITY>
```

```
<UTILITY NAME="QUIESCE">
  <MONITOR>
       <SYNTAX REMOVE="WRITE %"/>
  </MONITOR>
</UTILITY>
```

Related concepts:

"Example policy to invoke the Utility Monitor" on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

Related reference:

"<MONITOR>" on page 139
A element <MONITOR> identifies a set of syntax rules by which to evaluate the specified DB2 utility. This element is required when invoking the Utility Monitor and can only be specified once per <UTILITY> element.

<USE_PRACTICE>:

A <USE_PRACTICE> element specifies the name of the predefined practice to be used when evaluating the policy rules within the element <DB2SYSTEM>, and can only be specified once.

Used by elements

<DB2SYSTEM>

Child elements

No other elements.

Attributes

The following table describes the attributes that you can specify for the <USE_PRACTICE> element.

Table 20. <USE_PRACTICE> attributes

Attribute	Default Value	Required?
NAME Specifies the name of a practice (as defined in a <PRACTICE> element) that you want to assign to a DB2 subsystem. This practice will used when evaluating the policy rules. Wildcards are *not* permitted.	None	Yes

Example

```
<DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY">
   <USE_PRACTICE NAME="NORMAL_STANDARDS"/>
      <INCLUDE>
        <RULE UTILITY_USERID="%"/>
      </INCLUDE>
</DB2SYSTEM>
```

Related concepts:

"Example policy to invoke the Utility Monitor" on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

Related reference:

"<DB2SYSTEM>" on page 131
A <DB2SYSTEM> element identifies a DB2 subsystem for which to perform DSNUTILB intercept processing to block and cancel threads or to monitor DB2 utilities. If you specify a generic wildcard pattern as its attribute value, this element can identify multiple DB2 subsystems. You must specify one or more <DB2SYSTEM> elements within the <POLICY> section. You cannot specify a <DBSYSTEM> element in a <RULESET> section.

<USE_RULESET>:

A <USE_RULESET> element assigns a ruleset that you defined for thread-cancelation processing to a DB2 subsystem.

Used by elements

You can specify zero or more <USE_RULESET> elements under a <DB2SYSTEM> element in the <POLICY> section. The <USE_RULESET> element is optional. Instead of or in addition to rulesets, you can define individual <EXCLUDE> and <INCLUDE> rules for a DB2 subsystem.

<DB2SYSTEM>

Child elements

No other elements.

Attributes

The following table describes the only attribute that you can specify for the <USE_RULESET> element:

Table 21. <USE_RULESET> attributes

Attribute		Default Value	Required?
NAME	Specifies the name of a ruleset (as defined in a <RULESET> element in the same policy) that you want to assign to a DB2 subsystem. A ruleset is a collection of rules that identify the DB2 utilities, users, or objects for which to perform DSNUTILB intercept processing. Wildcards are *not* permitted.	None	Yes

Tip: To specify a ruleset name that is longer than the 80-column line length, type the name value up to column 80 and then continue it in column 1 of the next line. No special line-continuation characters are required. However, you cannot split the closing XML characters /> at the end the USE_RULESET definition.

Example

The following example assigns the ruleset "DB12RULES" to the DB2 subsystem "DB12." (The ruleset must be defined in a <RULESET> section of the same policy.) It also includes a specific <INCLUDE> rule that indicates DSNUTILB processing will be performed only for utility operations that are run under the user ID of "DBA1." If the referenced ruleset includes rules that conflict with this <INCLUDE> rule, the <INCLUDE> rule takes precedence because it occurs after the <USE_RULESET> reference in the policy. If the <USE_RULESET> reference occurred after the <INCLUDE> rule, the ruleset would take precedence.

```
<POLICY>
  <DB2SYSTEM SSID="DB12">
    <USE_RULESET NAME="DB12RULES"/>
    <INCLUDE>
      <RULE UTILITY_USERID="DBA1"/>
    </INCLUDE>
  </DB2SYSTEM>
</POLICY>
```

Related reference:

"<DB2SYSTEM>" on page 131
A <DB2SYSTEM> element identifies a DB2 subsystem for which to perform DSNUTILB intercept processing to block and cancel threads or to monitor DB2 utilities. If you specify a generic wildcard pattern as its attribute value, this element can identify multiple DB2 subsystems. You must specify one or more <DB2SYSTEM> elements within the <POLICY> section. You cannot specify a <DBSYSTEM> element in a <RULESET> section.

<UTILITY>:

The element <UTILITY>, when used in conjunction with the attribute NAME, identifies the DB2 utility that DB2 UET is to evaluate when DSNUTILB is invoked to verify whether the Practice rule should be enforced.

Used by elements

The <UTILITY> element is required to invoke the Utility Monitor. It can be defined multiple times within the element <PRACTICE>.

<PRACTICE>

Child elements

<MONITOR>

Attributes

The following table describes the attributes that you can specify for the <UTILITY> element.

Table 22. <UTILITY> attributes

Attribute	Default value	Required?
NAME Specifies name of the DB2 utility. Values: • CHECK_DATA • CHECK_INDEX • CHECK_LOB • COPY • COPYTOCOPY • DIAGNOSE • EXEC_SQL_ALTER • EXEC_SQL_DROP • LISTDEF • LOAD • MERGECOPY • MODIFY_RECOVERY • MODIFY_STATISTICS • OPTIONS • QUIESCE • REBUILD_INDEX • RECOVER • REORG_INDEX • REORG_INDEXSPACE • REORG_TABLESPACE • REPAIR • REPORT • RUNSTATS • STOSPACE • TEMPLATE • UNLOAD Wildcards are *not* permitted.	None	Yes
REPLACE Controls when to swap a DB2 UNLOAD utility with HPU. The only valid value is HPU.	HPU	No

Example

```
<PRACTICE NAME="STANDARDS_1">
       <UTILITY NAME="UNLOAD" REPLACE="HPU"/>
              .
              .
       </UTILITY>
</PRACTICE>
```

Related concepts:

Chapter 10, “Substituting HPU for the DB2 UNLOAD utility,” on page 295
DB2 UET can substitute the IBM DB2 High Performance Unload for z/OS (HPU) product for the DB2 UNLOAD utility.

"Example policy to invoke the Utility Monitor" on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

Related reference:

"<POLICY>" on page 141
The <POLICY> element is required for all types of processing that you can perform by using the DSNUTILB intercept. The <POLICY> element identifies the DB2 subsystems on which to implement the enhancements for the LOAD and REORG TABLESPACE utility or intercept processing.

"<PRACTICE>" on page 142
A <PRACTICE> element enables the DSNUTILB intercept feature of DB2 UET to search the provided DB2 utility statements and perform specified actions.

"<MONITOR>" on page 139
A element <MONITOR> identifies a set of syntax rules by which to evaluate the specified DB2 utility. This element is required when invoking the Utility Monitor and can only be specified once per <UTILITY> element.

How intercept policy rules are processed

DB2 UET reads the DSNUTILB policy in sequence. Therefore, the order in which you define rules and rulesets for thread-cancelation processing is important. The order and type of rules can affect which threads (if any) are blocked and canceled.

Processing logic

DB2 UET uses the following logic to process rules and rulesets in an intercept policy:

- For intercept processing to occur, at least one <INCLUDE> rule must be defined in the <POLICY> section or in a ruleset that is referenced from the <POLICY> section. If no rules or rulesets are specified, or if only exclusion rules are specified, no threads will be blocked and canceled.
- You can specify multiple <RULE> elements for the same attribute or different attributes under an <EXCLUDE> or <INCLUDE> element. If you define multiple <RULE> elements that specify different values for the same attribute (for example, UTILITY_USERID), these rules are related to one another by the "or" connector (only one must match). If you define multiple <RULE> elements that specify different attributes (for example, UTILITY_USERID, UTILITY_COMMAND, and DATABASE), these rules are related to another by the "and" connector (all must match).

The order of the RULEs

Rules and rulesets are processed in the order in which they are listed in the <POLICY> section, and this affects how the policy blocks and cancels threads. In general, you should specify <RULE> and <USE_RULESET> elements in the <POLICY> section in the order of increasing specificity (from least specific to most specific). For example, a RULE specifying which SHRLEVEL keywords to cancel should be listed within the element DB2SYSTEM before RULEs that contain explicitly-defined objects on which to block and cancel threads.

- If you specify multiple <USE_RULESET> elements under a <DB2SYSTEM> element to reference multiple <RULESET> definitions, the <RULESET> definitions will be processed in sequence, based on the order in which they are referenced by the <USE_RULESET> elements.

- If a referenced <RULESET> definition contains multiple <EXCLUDE> and <INCLUDE> rules, these rules will be processed in sequence based on the order in which they are listed within the <RULESET> element.
- If you define multiple <EXCLUDE> and <INCLUDE> rules directly under a <DB2SYSTEM> element in the <POLICY> section, these rules will be processed in sequence based on the order in which they are listed.
- If you specify both ruleset references and individual <EXCLUDE> and <INCLUDE> rules under a <DB2SYSTEM> element, the rules that are specified in the referenced rulesets and the rules that are specified individually are processed based on the order in which the <USE_RULESET> references and the individual <EXCLUDE> and <INCLUDE> rules are listed.

RULESETs and RULEs defined within each DB2 subsystem

Where multiple RULESETs exist that define the same object type and object value, only one RULESET should be included per DB2 subsystem.

For example, in "Example DSNUTILB intercept policies" on page 169, the RULESETs ACCOUNTS_PAYABLE and WEEKLY_MAINTENANCE both specify a RULE against a TABLESPACE containing the value ABPQDB01. Therefore it is recommended that only one RULESET be defined for inclusion or exclusion in the element DB2SYSTEM. Or, if one RULESET will be included and the other excluded, it is recommended that the more generic RULESET WEEKLY_MAINTENANCE be defined first, and the more specific RULESET ACCOUNTS_PAYABLE be defined last.

Unpredictable results may occur if both RULESETs are defined for either inclusion or exclusion within the same DB2SYSTEM element.

RULESETs and RULEs of the same object type

RULESETs that contain rules of the same object type use an "either / or" logic, whereas RULESETs that contain rules of different object types use "and" logic.

For example, a RULESET that contains multiple rules of the object type DATABASE uses the "either / or" logic. At least one of the objects within the RULESET must match the object defined in a DB2 utility in order for the ACTION to be invoked. However, if a RULESET contains rules of different object types, then all criterion within the RULESET must match a running utility, since different object types are ANDed together.

Matching RULEs or PRACTICE rules with delimited table names

Quotation marks are handled in a specific way within XML. When using RULEs or PRACTICE rules to match against a delimited table name to invoke an ACTION, the text string **&quot**; must be used. This string indicates to XML that a quotation mark should be used when matching the text string value contained within the element or attribute against the table name in a DB2 utility that is executed at the table level.

For example, if a LOAD utility contains a delimited table name in the syntax, such as "CSJENN"."ABPTB1", a RULE or PRACTICE rule would need to be defined like the following in order to match the delimited object name in the DB2 utility. To define a RULE to match a delimited table name, use the following syntax.

```
<RULE TABLE="&quot;CSJENN&quot;.&quot;ABPTB1&quot;"/>
```

The resulting table name within the RULE that will be matched against the table name in a DB2 utility will be “CSJENN”.“ABPTB1”. To define a PRACTICE rule to match a delimited table name, use the following syntax.

```
<SYNTAX VALUE="&quot;CSJENN&quot;.&quot;ABPTB1&quot;"
     SUBSTITUTE="&quot;CSJENN&quot;.&quot;ABPTEST1&quot;"/>
```

The resulting table name that will be matched within a DB2 utility will be “CSJENN”.“ABPTB1”. The value that will substituted into the DB2 utility will be “CSJENN”.“ABPTEST1”.

The following example presents a portion of the <POLICY> section that defines rules for all DB2 subsystems that have SSIDs beginning with “DB2”. The first exclusion rule is the most generic rule and is processed first. It prevents the DSNUTILB intercept from performing thread-cancelation processing for all utility users. The subsequent inclusion rules are more specific. They allow thread-cancelation processing whenever the “DBA1” user or “DBA2” user runs a utility against the “PAYROLL” database. If the <EXCLUDE> rule had followed the <INCLUDE> rules, no intercept processing would occur for any utility user.

```
<DB2SYSTEM SSID="DB2%">
  <EXCLUDE>
    <RULE UTILITY_USERID="%"/>
  </EXCLUDE>
  <INCLUDE>
    <RULE UTILITY_USERID="DBA1"/>
    <RULE UTILITY_USERID="DBA2"/>
    <RULE DATABASE="PAYROLL"/>
  </INCLUDE>
</DB2SYSTEM>
```

Creating a DSNUTILB intercept policy

If you plan to use the DSNUTILB intercept to implement the enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads, you must define an intercept policy in XML. The policy specifies the DB2 subsystem or subsystems on which to perform intercept processing and optionally the rules for selecting the threads to block and cancel.

Before you begin

Before you begin defining a policy, read the topics on the valid XML elements and attributes that you can use and review the example policies. See “Reading and understanding the policy” on page 131 and “Example DSNUTILB intercept policies” on page 169.

About this task

You can create only a single policy for a DB2 UET started task. This policy will define all intercept processing that you want to perform with the started task. The policy can be defined in a PDS member or a sequential file. It is recommended that you use the sample policy member *abpid*PLCY (where *abpid* is the value that you specified in Tools Customizer) as a starting point. The Tools Customizer created this member for your use in the *hlq.mlq*.SABPSAMP library.

Procedure

1. Customize the *abpid*PLCY member in the *hlq.mlq*.SABPSAMP library by using the ISPF editor. Use only the XML elements and attributes that are valid for the intercept policy. You will not be able to use the sample policy "as is." To implement the DB2 UET enhancements for the LOAD utility or the REORG

TABLESPACE utility, you need only to specify the DB2 subsystem or subsystems on which to perform the enhanced processing. To perform intercept processing, you must specify one or more DB2 subsystems and also define the rules or rulesets for selecting the DB2 utilities, users, or objects for intercept processing.

Important: Make sure that you customize the *abpid*PLCY member rather than the ABPPLCY member on which it is based.

2. Save your customized policy member.

What to do next

Ensure that your customized policy member is correctly specified in the ABPPLCY DD statement of your started task PROC (*abpid*PROC). Also, ensure that the DTD DD statement in the PROC specifies the partitioned data set (a PDS or PDSE) that contains both the DTD for the intercept policy and the DTD for the started task initialization options.

Related concepts:

"Example DSNUTILB intercept policies" on page 169
Review the following example DSNUTILB intercept policies to learn how you can create a policy to meet your intercept processing needs.

"Example policy that details RULE and RULESET sections" on page 252
In general, RULEs should be defined from the least specific to the most specific. The rules in the <RULESET> section are read and processed from top to bottom. Therefore, the order in which rules are defined is important.

"Task flow for blocking and canceling threads" on page 183
This task flow presents the high-level tasks that you will need to perform to block and cancel threads by using the DSNUTILB intercept.

Policy validation algorithm

The policy validation algorithm of the DB2 UET supplies the logic to evaluate policy rules against environmental and DB2-specific constructs in order to determine actions that may be taken by DB2 UET.

Policy rules

An invocation of the DB2 utility program DSNUTILB constitutes a unit of work for the DSNUTILB intercept interface of DB2 UET. This interface receives control from a call by DSNUTILB to the DSNUTILF exit routine. A key feature of the DSNUTILB intercept is DB2 UET ability to selectively apply actions to the execution of DB2 utility programs based on a user-supplied policy.

The policy is specified in an XML document that is parsed by the DB2 UET server as part of the server's initialization. The policy has a number of features, but this section will focus on the concept of a policy rule. A rule is defined as a set of elements and their corresponding attributes specified within either an INCLUDE or an EXCLUDE statement that dictate if a thread cancel and block is to occur. The following example of a policy rule contains a RULE element with an attribute of TABLESPACE. The value of the TABLESPACE attribute is indicated by the two-part <database>.<tablespace> name "MYDBASE.MYTSPACE."

```
<INCLUDE>
   <RULE TABLESPACE="MYDBASE.MYTSPACE"/>
</INCLUDE>
```

The DB2 UET provides a variety of rule elements to provide flexibility in defining when thread cancelations and blocks should occur. As a result, a rule can contain any combination and number of RULE elements. Policy rules can be named by the user by utilizing the RULESET element. By default, policy rules are unnamed unless a RULESET element is defined to name the rule or set of rules. Note, however, that a policy rule is a collection of RULE elements that can be INCLUDEd or EXCLUDEd. The following example is of a policy containing three rules.

```
<INCLUDE>
   <RULE UTILITY_USERID="ADMIN00%"/>
   <RULE UTILITY_USERID="DB2SYSAM"/>
   <RULE UTILITY_JOBNAME="PAYR%"/>
</INCLUDE>
<INCLUDE>
   <RULE UTILITY_COMMAND="REORG_INDEX"/>
   <RULE INDEX="ABPTST02.TST02TB1_IX1"/>
   <RULE INDEX="ABPTST02.TST02TB1_IX2"/>
</INCLUDE>
<EXCLUDE>
   <RULE UTILITY_USERID="ADMIN004"/>
</EXCLUDE>
```

The user IDs ADMIN00% and DB2SYSAM are being included in the rule along with the job name matching the pattern PAYR%. That is, when the user IDs and the job name are matched with activity on the DB2 subsystem, thread cancel and block activity will be invoked for those activities associated with the user IDs and job name.

The utility user ID ADMIN004 is being excluded from the policy. That is, when the user ID is matched with a utility running on the DB2 subsystem, thread cancel and block activity will not be invoked for that utility associated with the user ID.

The specification of INCLUDE and EXCLUDE rules allow the policy administrator to customize how DB2 UET acts upon utility execution. INCLUDE rules allow action while EXCLUDE rules prohibit action.

RULE elements

A policy rule can contain any number of RULE elements. An element is an attribute upon which a value can be associated and evaluated. There are 13 RULE elements that can currently be specified within DB2 UET. RULE elements are matched by the policy validation algorithm with target strings specified in the utility job runtime environment or control statements.

For more information on Utility environment, Utility syntax, and Utility object RULE elements, see“<RULE>” on page 144.

<RULE> elements coded within a policy rule determine whether or not the policy action will be enforced. At the end of each policy validation, an action decision is made based on the policy rules plus the utility job environment and control statements.

Policy actions

As each rule is analyzed, a result of ACTION ON, ACTION OFF or NO ACTION is determined.

- ACTION ON – A policy rule is successfully matched with utility match targets.

- ACTION OFF – A policy rule contains RULE elements that do not successfully match utility match targets.
- NO ACTION - None of the RULE elements in the policy rule have corresponding utility match targets to match against.

An example of an action that DB2 UET may take on behalf of a utility execution is to cancel existing threads holding locks and/or claims against a particular object and block new threads before the utility executes. After the utility executes, DB2 UET then allows new threads to access the objects. The policy rules in effect at the time of the DSNUTILB intercept by DB2 UET determine whether or not the utility job would have this action taken by DB2 UET on its behalf. It is the responsibility of the policy validation algorithm to make this determination.

Policy logic

Logic is applied to rules contained within a policy document and to each RULE element within a policy rule. The relationship between the RULE elements within a policy rule determines the action DB2 UET will take. By default an action is turned off.

The following policy statement will instruct DB2 UET to intercept utility execution on DB2 subsystem D81E but no actions will occur since no INCLUDE rules are specified.

```
<DB2SYSTEM SSID="D81E">
<DB2SYSTEM>
```

Defining different RULE elements

Any number of RULE elements can be specified within one policy rule. When multiple RULE elements are specified, all of the distinct RULE elements must match in order to turn the action ON.

The following example contains an INCLUDE rule with two distinct RULE elements; USERID and JOBNAME.

```
<INCLUDE>
   <RULE UTILITY_USERID="PACKMAN"/>    <---
                                         |-- AND
   <RULE UTILITY_JOBNAME="EATDOT"/>    <---
</INCLUDE>
```

A utility job run by user PACKMAN with a jobname of EATDOT will match both of the RULE elements in the policy rule resulting in turning an action ON. If either of the RULE elements fails to match either the user ID or the job name, then the action will be turned OFF.

If none of the RULE elements in the rule relate to the utility job, a result of NO ACTION is returned in the SYSPRINT and the rule is not considered in the final evaluation. For more information, see the usage examples.

Defining the same RULE elements

The same RULE element can be specified more than once within an INCLUDE statement. In the following example, either the user ID PACKMAN or MSPKMAN must match the utility job user ID in order for the action to turn ON.

```
<INCLUDE>
   <RULE UTILITY_USERID="PACKMAN"/>    <---
                                        |-- OR
   <RULE UTILITY_USERID="MSPKMAN"/>    <---
</INCLUDE>
```

The same RULE element can be specified more than once with separate INCLUDE statements, effectively overriding previously defined elements of the same type. That is to say, the order that policy rules are specified affects whether an action is turn on or off.

In the following example, the first UTILITY_ID rule includes actions on any utility containing a utility ID starting with “UTIL_PAY”. The next UTILITY_ID rule restricts actions to only utilities with a utility id of “UTIL_PAYROLL”. The second INCLUDE rule overrides the first.

```
<INCLUDE>
   <RULE UTILITY_ID="UTIL_PAY%"/>
</INCLUDE>
<INCLUDE>
   <RULE UTILITY_ID="UTIL_PAYROLL"/>
</INCLUDE>
```

Distinct RULE elements in a policy rule are logically combined with an AND relationship. That is to say all defined string values must match those string values discovered in the utility in order to turn the action ON. Duplicate RULE elements in a policy rule are logically combined with an OR relationship. That is to say any defined string value can match those string values discovered in the utility in order to turn the action ON.

The order of the RULES

In the following example, a utility job with the job name ABPJOB00 run by user USER01 will turn the action ON. The second INCLUDE rule overrides the ACTION OFF result of the previous EXCLUDE rule, because it is the last rule in the list.

```
<EXCLUDE>
   <RULE UTILITY_JOBNAME="ABPJOB00"/>
</EXCLUDE>
<INCLUDE>
   <RULE UTILITY_USERID="USER01"/>
</INCLUDE>
```

Diagnostic output

Diagnostic information is provided by the policy validation function if the ABPRDIAG DD statement is included in the DB2 UET started task JCL. By default, the ABPRDIAG DD is defined in the started task JCL, but can be removed at any point after installation. As the policy validation algorithm is executing, status information is written to ABPRDIAG that can be helpful in understanding how the policy rules were evaluated.

For example, if a REORG utility with the job name ABPJOB00 was executing against table space ABPTST01.TST02TS1 using the policy rules in the following example, then the diagnostic information would be written to ABPRDIAG.

```
<INCLUDE>
   <RULE UTILITY_JOBNAME="ABPJOB00"/>
   <RULE TABLESPACE="ABPTST02.TST02TS1"/>
</INCLUDE>
<EXCLUDE>
```

```
   <RULE UTILITY_ID="VFHTEST"/>
</EXCLUDE>

(1) INCLUDE RULE 00000001
(2)   RULE ELEMENT - UIILITY_JOBNAME              SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
(3)     P: ABPJOB00
(4)     F: ABPJOB00
(5)   RULE ELEMENT - TABLESPACE : DATABASE        SEQUENCE NUMBER: 00000002  MATCH    CONTINUE
        P: ABPTST02
        F: ABPTST02
(6)   RULE ELEMENT - TABLESPACE : SPACE           SEQUENCE NUMBER: 00000002  MATCH    ACTION ON
        P: TST02TS1
        F: TST02TS1
(7) INCLUDE RULE 00000001      *****   ACTION ON    *****

(8) EXCLUDE RULE 00000002
      RULE ELEMENT - UIILITY_ID                   SEQUENCE NUMBER: 00000003  MATCH    ACTION OFF
        P: VFHTEST
        F: VFHTEST
(9) EXCLUDE RULE 00000002      *****   ACTION OFF   *****
```

The following description of the ABPRDIAG contents helps illustrate the evaluation logic that was applied to the policy Rules from the previous example.

1. This is the beginning of the policy rule message that identifies the rule type and rule number. The rule number is determined by the position of the rule in the policy document. That is, the first rule in the policy will be numbered rule #1. The second rule in the policy will be numbered rule #2, and so on.
2. RULE element message that identifies:
   a. The RULE element name as it appears in the policy.
   b. Sequence number that corresponds to the sequence provided in message ABPS0833I (found in the DB2 UET started task SYSPRINT log).
   c. MATCH/NOMATCH identifies if the pattern that was provided in the rule in item (3) matches the actual value found in item (4).
   d. ACTION status ON/OFF or CONTINUE if the RULE element analysis is continued with another compare step.
3. P: displays the string pattern that was provided in the rule.
4. F: displays the string value that was found.
5. Another RULE element message that identifies:
   a. The RULE element name as it appears in the policy.
   b. Sequence number that corresponds to the sequence provided in message ABLS0833I (found in the DB2 UET started task SYSPRINT log).
   c. MATCH/NOMATCH identifies if the pattern that was provided in the rule matches the actual value found.
   d. ACTION status ON/OFF or CONTINUE if the RULE element analysis is continues with another compare step.

   In this rule, the attribute is TABLESPACE, which is composed of a two-part DB2 object name. Since the string pattern that was provided for the table space qualifier in the rule matched the string value that was found for the table space qualifier, the action status is CONTINUE, which means it will continue to evaluate the remaining attribute of the rule before determining if the ACTION is ON or OFF.
6. The analysis for the table space name in the RULE element TABLESPACE. Note that the sequence number for both parts is the same because the table space name is evaluated after the table space qualifier is completed.
7. This is the final message that displays the analyzed ACTION for the rule that contains strings for the attributes JOBNAME and TABLESPACE.
8. This is the beginning of another policy rule message for the attribute UTILITY_ID. After being evaluated, this rule results in the ACTION OFF since

the rule was created to be EXCLUDEd if the RULE element string value that was provided matches the string value that was found.

9. This is the final message that displays the analyzed ACTION for the rule UTILITY_ID.

Policy use case examples:

This section provides examples that illustrate various use cases.

Utility control statements can direct a utility to act on multiple DB2 objects. The following example utility statement will be evaluated against the following policy rule:

```
CHECK DATA TABLESPACE ABPTSTD1.ABPTSTS1
           TABLESPACE ABPTSTD3.ABPTSTS3
           TABLESPACE ABPTSTD2.ABPTSTS2

<INCLUDE>
   <RULE TABLESPACE="ABPTSTD2.ABPTSTS2"/>
   <RULE TABLESPACE="ABPTSTD1.ABPTSTS1"/>
   <RULE TABLESPACE="ABPTSTD3.ABPTSTS3"/>
</INCLUDE>
```

Because all table space objects referenced in the utility statement are also referenced in the rule, and the rule is specified to be included, each object within the utility is evaluated against the objects within the rule, and the resultant analysis by the policy validation algorithm is ACTION ON.

The resultant diagnostic output follows:

```
INCLUDE RULE 00000001
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: ABPTSTD2
    F: ABPTSTD1
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000002  MATCH    CONTINUE
    P: ABPTSTD1
    F: ABPTSTD1
  RULE ELEMENT - TABLESPACE : SPACE         SEQUENCE NUMBER: 00000002  MATCH    ACTION ON
    P: ABPTSTS1
    F: ABPTSTS1
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: ABPTSTD2
    F: ABPTSTD3
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000002  NOMATCH  ACTION OFF
    P: ABPTSTD1
    F: ABPTSTD3
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000003  MATCH    CONTINUE
    P: ABPTSTD3
    F: ABPTSTD3
  RULE ELEMENT - TABLESPACE : SPACE         SEQUENCE NUMBER: 00000003  MATCH    ACTION ON
    P: ABPTSTS3
    F: ABPTSTS3
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  MATCH    CONTINUE
    P: ABPTSTD2
    F: ABPTSTD2
  RULE ELEMENT - TABLESPACE : SPACE         SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
    P: ABPTSTS2
    F: ABPTSTS2
INCLUDE RULE 00000001      *****   ACTION ON    *****
```

If one of the objects referenced in the utility statement is not defined in the rule, then the rule will not match for the utility and the resultant analysis for the policy validation algorithm is ACTION OFF. That is to say, because there are multiple table spaces defined within one utility, all table spaces must be matched in the rule in order for the action to be ON.

Using the previous utility statement but omitting the rule element for table space ABPTSTD3.ABPTSTS3, the resultant analysis by the policy validation algorithm is ACTION OFF. This is because the third table space listed within the utility does not have a rule element defined within the policy rules.

```
<INCLUDE>
   <RULE TABLESPACE="ABPTSTD2.ABPTSTS2"/>
   <RULE TABLESPACE="ABPTSTD1.ABPTSTS1"/>
</INCLUDE>
```

The resultant diagnostic output follows:

```
INCLUDE RULE 00000001
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: ABPTSTD2
    F: ABPTSTD1
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000002  MATCH    CONTINUE
    P: ABPTSTD1
    F: ABPTSTD1
  RULE ELEMENT - TABLESPACE : SPACE         SEQUENCE NUMBER: 00000002  MATCH    ACTION ON
    P: ABPTSTS1
    F: ABPTSTS1
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: ABPTSTD2
    F: ABPTSTD3
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000002  NOMATCH  ACTION OFF
    P: ABPTSTD1
    F: ABPTSTD3
INCLUDE RULE 00000001      *****   ACTION OFF   *****
```

The result is ACTION OFF because all three table spaces defined within the utility were not present in the rule. When repeated elements are used within the same rule, they are ANDed together, such that all objects within one rule must be evaluated as a group for the action to be evaluated as ON.

Multiple utilities may be specified in a single call to the DSNUTILB control program. The following example utility statement will be evaluated against the following policy rule:

```
CHECK DATA TABLESPACE (ABPTST02.TST02TS1)
CHECK INDEX (ABPTST02.TST02TB1_IX1)

<INCLUDE>
   <RULE TABLESPACE="ABPTST02.TST02TS1"/>
</INCLUDE>
<INCLUDE>
   <RULE INDEX="ABPTST02.TST02TB1_IX1"/>
</INCLUDE>
```

In this example, each INCLUDE rule relates to a specific object. The table space rule will be evaluated against the object in the CHECK DATA utility. The index rule will not be evaluated against the object in the CHECK DATA utility, because it is the wrong object type.

Likewise, when the CHECK INDEX utility is run, the index rule will be evaluated against the object in the CHECK INDEX utility. The table space rule will not be evaluated against the object in the CHECK INDEX utility, because it is the wrong object type.

```
INCLUDE RULE 00000001
  RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  MATCH    CONTINUE
    P: ABPTST02
    F: ABPTST02
  RULE ELEMENT - TABLESPACE : SPACE         SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
    P: TST02TS1
    F: TST02TS1
INCLUDE RULE 00000001      *****   ACTION ON    *****
```

```
INCLUDE RULE 00000002
  RULE ELEMENT - INDEX                    SEQUENCE NUMBER: 00000002  -----    NO ACTION
INCLUDE RULE 00000002      *****   ACTION ON    *****
```

In the previous example, the Rule Element TABLESPACE is evaluated against the two-part DB2 object name. The table space qualifier is evaluated first, then the table space name is evaluated. The resultant action is ON because the two-part name of the object name in the utility matches the value string provided in the rule.

Notice the Rule Element INDEX contains dashes ('----') in the Match column along with the message NO ACTION. This indicates that the rule was not evaluated because the element value was INDEX and not TABLESPACE. Because CHECK DATA occurs at the TABLESPACE level, the rule containing the INDEX element is ignored.

Some utilities can target table spaces, indexspaces, or index partitions through the DSNUM or PART attribute. This specification can be matched in the DB2 UET policy using the PART(part_spec) option of the TABLESPACE, INDEXSPACE, or INDEX RULE elements. The PART option of the policy allows you to specify any combination of partitions within a given space. There can be any number of single partitions or partition ranges up to a total limit of 256 bytes between the quotes. Partition ranges can be annotated by using a colon.

The following example utility statement that defines partitions 5, 6, 7, and 8 of table space ABPTST02.TST02TS1 will be evaluated against the following policy rule that defines partitions 1, 4, 5, 6, 7 and 8 of table spaces that match wildcard pattern _BP%.TST%:

```
REORG TABLESPACE ABPTST02.TST02TS1 PART 5:8
      NOSYSREC
```

```
<DB2SYSTEM SSID="D81E">
<INCLUDE>
   <RULE TABLESPACE=="_BP%.TST%" PART="1,4:8"/>
</INCLUDE>
</DB2SYSTEM>
```

The resultant diagnostic output follows:

```
SESSION: 00000001    STEP: 00000001  ---  BLOCK_AND_CANCEL_THREADS
UTILITY: REORG_TABLESPACE
RULE ELEMENT - TABLESPACE : DATABASE      SEQUENCE NUMBER: 00000001  MATCH    CONTINUE
      P: _BP%
      F: ABPTST02
    RULE ELEMENT - TABLESPACE : SPACE         SEQUENCE NUMBER: 00000001  MATCH    CONTINUE
      P: TST%
      F: TST02TS1
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
      P: 1,4:6,7,8
      F: 00000005
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
      P: 1,4:6,7,8
      F: 00000005
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
      P: 1,4:6,7,8
      F: 00000006
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
      P: 1,4:6,7,8
      F: 00000006
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
      P: 1,4:6,7,8
      F: 00000007
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
      P: 1,4:6,7,8
      F: 00000007
    RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
      P: 1,4:6,7,8
```

```
    F: 00000007
  RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: 1,4:6,7,8
    F: 00000008
  RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: 1,4:6,7,8
    F: 00000008
  RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  NOMATCH  ACTION OFF
    P: 1,4:6,7,8
    F: 00000008
  RULE ELEMENT - TABLESPACE : PARTITIONS    SEQUENCE NUMBER: 00000001  MATCH    ACTION ON
    P: 1,4:6,7,8
    F: 00000008
INCLUDE RULE 00000001      *****   ACTION ON    *****
```

In all examples in this section, if the PART option is not specified in the RULE element, then a match is not attempted at the partition level, and the current action is carried forward.

The DSNUM utility syntax option is matched to the policy rule parameter part_spec using the following guidelines:

- DSNUM ALL - All partitions of the space must be specified in the policy rule parameter part_spec in order for the RULE element to match.
- DSNUM integer - The DSNUM partition must match one of the partitions defined in the policy rule parameter part_spec.

The PART utility syntax option is matched to the policy rule parameter part_spec using the following guidelines:

- PART integer - The PART partition must match one of the partitions defined in the policy rule parameter part_spec.
- PART integer1:integer2 - The range of partitions must match the range given in the policy rule parameter part_spec.

Each partition that is defined for the utility (in this example, partitions 5, 6, 7, and 8) is consecutively compared with each partition range that is defined in the policy (in this example, partition 1, then range 4-8). When a match in any one range is detected, then ACTION is set to ON, and the next partition check starts. The matching partition is not compared with the remainder of the policy partition definitions. If any one partition does not match at least one range that is defined in the policy, then ACTION is set to OFF, and the remaining partitions are not checked.

Example DSNUTILB intercept policies

Review the following example DSNUTILB intercept policies to learn how you can create a policy to meet your intercept processing needs.

Example 1: Policy that does not require rules and rulesets

The following example policy contains only the elements that are needed to implement the additional options for the LOAD utility or the automatic sizing and creation of mapping tables and mapping-table indexes for the REORG TABLESPACE utility on a single DB2 subsystem. No rules or rulesets are included because these enhancements do not use rules and rulesets.

This example policy can also be used to block and cancel threads, because the corresponding ACTION element is not present, and it therefore defaults to BLOCK_AND_CANCEL_THREADS.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE OPTIONS SYSTEM "DD:DTD(ABPDTDPL)">
<DSNUTILB_INTERCEPT>
 <POLICY>
  <DB2SYSTEM SSID="DBP1"/>
 </POLICY>
</DSNUTILB_INTERCEPT>
```

Example 2: Policy to define two rulesets, each specific to a particular DB2 subsystem

The following example policy defines two rulesets, each specific to a particular DB2 subsystem, for selecting the threads to block and cancel for certain DB2 utilities. On the DBP1 subsystem, threads can be blocked and canceled for COPY, LOAD, and REORG TABLESPACE operations that are performed by a person who has a utility user ID beginning with "DBA" and that is performed against objects other than tables that have the creator ID of "ABPUDB01". For COPY and REORG TABLESPACE operations, the objects must be in the "ABPUDB1" database. However, for LOAD operations, the objects can be in any database. The DATABASE rule is ignored for LOAD operations because the LOAD utility operates on tables and the DATABASE rule is matched only against objects with two-part names that include a database name (that is, index space or table space names). On the DBP2 subsystem, threads can be blocked and canceled for REORG_TABLESPACE operations that are performed by a person who has the specific utility user ID of "DBADMIN" and that are performed against any table space in the database "ABPUDB1A". In addition to being used for thread blocking and cancelation, this policy can also be used to identify the DB2 subsystems on which to implement the additional LOAD options (if specified) and the REORG TABLESPACE mapping-table enhancements.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE OPTIONS SYSTEM "DD:DTD(ABPDTDPL)">
<DSNUTILB_INTERCEPT>
 <RULESET NAME="DBP1_RULES">
  <INCLUDE>
   <RULE UTILITY_USERID="DBA%"/>
   <RULE UTILITY_COMMAND="COPY"/>
   <RULE UTILITY_COMMAND="LOAD"/>
   <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
   <RULE DATABASE="ABPUDB1"/>
  </INCLUDE>
  <EXCLUDE>
   <RULE TABLE="ABPUDB01.%"/>
     </EXCLUDE>
 </RULESET>
 <RULESET NAME="DBP2_RULES">
  <INCLUDE>
   <RULE UTILITY_USERID="DBADMIN"/>
   <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
   <RULE TABLESPACE="ABPUDB1A.%"/>
  </INCLUDE>
 </RULESET>
 <POLICY>
  <DB2SYSTEM SSID="DBP1">
   <USE_RULESET NAME="DBP1_RULES"/>
        <INCLUDE>
         <RULE UTILITY_ID="%">
        </INCLUDE>
  </DB2SYSTEM>
  <DB2SYSTEM SSID="DBP2">
   <USE_RULESET NAME="DBP2_RULES"/>
        <INCLUDE>
         <RULE UTILITY_ID="%">
```

```
       </INCLUDE>
  </DB2SYSTEM>
 </POLICY>
</DSNUTILB_INTERCEPT>
```

Example 3: Policy to define several rulesets for different user types

The following example policy defines several rulesets for different user types. It also defines a separate exclusion rule within the <POLICY> section to temporarily exclude certain tables that are being used for test purposes on one DB2 subsystem. In the <POLICY> section, rules are defined in general-to-specific order, as recommended. Based on this policy, thread-cancelation processing could occur on the DBP2 and DBP3 subsystems based on the rulesets and rules that are specified for these subsystems. No additional processing would occur on DBP1 subsystem because no rulesets or rules are defined for it. The utility-specific enhancements for the LOAD and REORG TABLESPACE utilities could be implemented on any or all of the subsystems: DBP1, DBP2, and DBP3 as long as rulesets or rules are defined for them.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE OPTIONS SYSTEM "DD:DTD(ABPDTDPL)">
<DSNUTILB_INTERCEPT>
 <RULESET NAME="DEFAULT_RULES">
  <!-- ------------------------------------------------------ -->
  <!-- Include all users with IDs beginning PD                -->
  <!-- ------------------------------------------------------ -->
  <INCLUDE>
       <RULE UTILITY_USERID="PD%"/>
  </INCLUDE>
 </RULESET>
 <RULESET NAME="INTERN_RULES">
  <!-- ------------------------------------------------------ -->
  <!-- Exclude student interns except when they are using     -->
  <!-- the INTERNDB database                                  -->
  <!-- ------------------------------------------------------ -->
  <EXCLUDE>
       <RULE UTILITY_USERID="PDINTRN%"/>
  </EXCLUDE>
   <INCLUDE>
       <RULE UTILITY_USERID="PDINTRN%"/>
       <RULE DATABASE="INTERNDB"/>
   </INCLUDE>
 </RULESET>
 <RULESET NAME="DBA_RULES">
  <!-- ------------------------------------------------------ -->
  <!-- Include database administrator DBA1 when a utility     -->
  <!-- runs against the PAYROLL or ACCTSREC database          -->
  <!-- ------------------------------------------------------ -->
  <INCLUDE>
       <RULE UTILITY_USERID="PDDBA1"/>
       <RULE DATABASE="PAYROLL"/>
       <RULE DATABASE="ACCTSREC"/>
  </INCLUDE>
  <!-- ------------------------------------------------------ -->
  <!-- Include database administrator DBA2 when a utility     -->
  <!-- runs against the ACCTSREC database                     -->
  <!-- ------------------------------------------------------ -->
  <INCLUDE>
       <RULE UTILITY_USERID="DBA2"/>
       <RULE DATABASE="ACCTSREC"/>
  </INCLUDE>
 </RULESET>
 <RULESET NAME="SYSADM_RULES">
  <!-- ------------------------------------------------------ -->
  <!-- Include all system administrators who have a SYSADM    -->
```

```
  <!-- user ID                                                  -->
  <!-- ------------------------------------------------------- -->
  <INCLUDE>
       <RULE UTILITY_USERID="SYSADM%"/>
  </INCLUDE>
 </RULESET>
 <POLICY>
  <DB2SYSTEM SSID="DBP1"/>
  <DB2SYSTEM SSID="DBP2">
      <USE_RULESET NAME="DBA_RULES"/>
      <USE_RULESET NAME="SYSADM_RULES"/>
      <EXCLUDE>
        <RULE TABLE CRT01.TST%/>
      </EXCLUDE>
  </DB2SYSTEM>
  <DB2SYSTEM SSID="DBP3">
      <USE_RULESET NAME="DEFAULT_RULES"/>
      <USE_RULESET NAME="INTERN_RULES"/>
  </DB2SYSTEM>
 </POLICY>
</DSNUTILB_INTERCEPT>
```

Related concepts:

"Example policy that details RULE and RULESET sections" on page 252
In general, RULEs should be defined from the least specific to the most specific. The rules in the <RULESET> section are read and processed from top to bottom. Therefore, the order in which rules are defined is important.

"Example policy to invoke the Utility Monitor" on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

"Example Practice rules" on page 249
Practice rules illustrate the flexibility of the Utility Monitor.

Related tasks:

"Creating a DSNUTILB intercept policy" on page 160
If you plan to use the DSNUTILB intercept to implement the enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads, you must define an intercept policy in XML. The policy specifies the DB2 subsystem or subsystems on which to perform intercept processing and optionally the rules for selecting the threads to block and cancel.

Chapter 5. Blocking and canceling DB2 threads

You can cancel active DB2 threads by using any of the DB2 UET interfaces. By canceling threads, you can free DB2 objects that are being held so that other applications or utilities can access them.

All of the interfaces support the following cancelation methods:

- A normal DB2 cancelation, which uses the DB2 -CANCEL THREAD command to cancel the thread.
- An escalated cancelation option that you can optionally use when a normal DB2 cancelation fails. An escalated cancelation issues the z/OS Cancel command to terminate the TSO user, started task, or batch job that is associated with the thread.

By default, backout processing is enabled for both normal and escalated thread cancelations.

Restriction:

- DB2 UET does not cancel the following threads:
  - Threads that have a thread token value of zero, to include CICS® protected and reusable threads.
  - The thread for a current TSO user connection to the DB2 UET ISPF interface
  - In DB2 Version 8, claims that do not hold locks will not be canceled. Only claims that hold locks will be canceled in DB2 Version 8.
- DB2 UET does not block threads on the DB2 catalog tables or directory tables. It will cancel threads, but will not block additional threads from coming active.

With any of the interfaces, you can specify criteria for selecting the threads to cancel. You can also use a pre-cancel exit, post-cancel exit, or both if you want to perform any additional processing before or after thread cancelations.

If you use the batch interface or DSNUTILB intercept, you can override backout processing, if necessary.

Your choice of interface and thread-cancelation configuration will depend on your reason for canceling threads. One interface might be more suitable than another for your situation. Use the following general guidelines:

Table 23. Choosing the interface to use for thread-cancelation

Goal	Recommended interface
Transparently block and cancel threads for DB2 online utilities	**DSNUTILB intercept** You will need to create an intercept policy that defines the utilities, users, or objects for which to cancel threads. Then, whenever DSNUTILB initiates the specified utility, the threads on the DB2 objects that the utility needs to access are automatically canceled. For more information, see "Blocking and canceling threads by using the DSNUTILB intercept" on page 180.
Cancel threads on an ad hoc basis	**ISPF interface** The ISPF interface provides an interactive interface from which you can filter active threads, cancel threads selectively, and view thread details. For more information, see "Canceling threads by using the ISPF interface" on page 185.
Ensure that applications, utilities, and DBA activities that are part of batch jobs have access to the DB2 objects that they need during the batch window	**batch interface** You can create a batch job step that both cancels existing threads and blocks new threads forming until after your application or utility completes. This feature helps prevent DB2 access problems and nighttime calls to your DBAs. For more information, see "Blocking and canceling threads by using the batch interface" on page 208.

Topics:

Related concepts:

"Blocking and canceling threads by using the DSNUTILB intercept" on page 180
You can use the DSNUTILB intercept to transparently block and cancel threads for the DB2 online utilities. The intercept can block and cancel threads for all DB2 utilities for which this function is appropriate.

"Canceling threads by using the ISPF interface" on page 185
The DB2 UET ISPF interface provides a central point from which you can interactively monitor and cancel DB2 threads that originate from various sources in

your environment.

"Blocking and canceling threads by using the batch interface" on page 208
The DB2 UET batch interface enables you to create a batch job step that cancels threads on the DB2 objects that an application or utility needs to access during batch processing. Optionally, this job step can also block new threads from forming on the objects that a utility needs until after the utility completes.

Examples and scenarios

Review the use cases and scenarios to learn how you could use the thread-block-and-cancel feature in a DB2 for z/OS environment.

Thread-cancelation by user role

Several types of users in a z/OS environment can benefit from using the thread-block-and-cancel feature both daily and occasionally. The following table summarizes some typical uses by user role:

Table 24. Summary of thread-block-and-cancel uses

User role	Thread-block-and-cancel uses	Frequency of use
Production control coordinator	To block and cancel threads to free DB2 resources that are needed by utilities and applications that will run as part of the production job stream	Daily
DB2 database administrator or system programmer	To block and cancel threads to free DB2 resources prior to performing database maintenance tasks such as a DB2 REORG, ALTER, UNLOAD, or DROP operation To cancel threads to terminate errant or long-running applications or jobs that are tying up DB2 objects that are required by other applications	Daily
Applications programmer	To block and cancel threads that originate from a test or in-house application that is not responding or that has an error	Occasionally

Scenario 1: Ensure that the LOAD utility can access a table to be loaded

Assume that you are a production control coordinator and need to update a table weekly during the batch window by loading the latest data from a sequential file. You plan to use the DB2 LOAD utility. To ensure that the LOAD utility can access the table to be loaded, you want to cancel existing threads on the table just before the utility runs. You can accomplish this task by using the batch interface:

1. Create a batch job step that includes at least one CANCEL_THREADS request that includes the TABLE *creator_id.table_name* parameter. This parameter identifies the table for which to cancel threads.
2. Add your job step to the batch job just *before* the LOAD utility job step. When the batch job runs, threads on the specified table will be canceled before the utility runs.

If the LOAD utility required exclusive access to the table, you could also code batch job steps to block new threads, as well as cancel existing threads, or you could have used the DSNUTILB intercept instead.

Scenario 2: Ensure that the COPY utility has immediate access to a table space

Assume that you are a DB2 DBA and need to create a full image copy of a table space for an application. You code a COPY utility job that includes the SHRLEVEL REFERENCE option because you want no users to update the table space during the copy operation. To ensure that the COPY utility has immediate access to the table space, you can block and cancel threads on the table space by using the DB2 UET DSNUTILB intercept. The intercept component intercepts the DSNUTILB program to transparently block and cancel threads on the table space when the COPY utility runs.

To use the intercept, you must create a valid DSNUTILB intercept policy. The policy must specify the DB2 system that contains the table space to be copied and one or more rules that determine whether to block and cancel threads for the utility execution. (You can define various types of rules, for example, rules that identify the specific table space, the COPY utility command, the SHRLEVEL option, or a particular user ID.) Immediately after DSNUTILB invokes the COPY utility, thread blocking and cancelation occurs transparently.

Scenario 3: Ensure that the REORG TABLESPACE utility has exclusive access to a table space during batch processing

Assume that you are a DB2 DBA and need to run the DB2 REORG TABLESPACE utility to reorganize a table space during batch processing. Because you want the REORG TABLESPACE utility to have exclusive access to the table space, you create a batch job step that cancels existing threads on the table space and also blocks new threads from forming on the table space until after the utility completes.

You must create two job steps: one to block and cancel threads, and one to allow new threads to form again.

1. In the EXEC statement for the block-and-cancel-threads job step, include `PARM='blocker_id,BLOCK_THREADS'` to enable thread blocking.
2. Specify a CANCEL_THREADS request with the TABLESPACE *database_name.tablespace_name* parameter. This parameter specifies the name of the table space that you need to reorganize and the database that contains it.
3. In the EXEC statement for the separate allow-threads job step, include `PARM='blocker_id,ALLOW_THREADS'` (where *blocker_id* is the same identifier that you specified in the BLOCK_THREADS job step). You do not need to add any CANCEL_THREADS requests to this job step.
4. Add the block-and-cancel-threads job step to the batch job just before the REORG utility job step.
5. Add the allow-threads job step just after the utility job step.

When you run the batch job, threads will be blocked and canceled immediately before the utility runs and will continue to be blocked until after the utility completes.

If the utility did not require exclusive access to the table space, you could define only one job step that canceled threads. This job step would include one or more CANCEL_THREADS requests but no PARM that specified a thread-blocker action on the EXEC statement.

Scenario 4: Cancel threads so that you can rebind the DB2 plan

Assume that you are a system programmer and notice that a DB2 subsystem has a performance problem. You run the DB2 RUNSTATS utility to produce a statistics report to help you determine the optimal access paths to DB2 objects. After running the utility, you attempt to rebind the DB2 plan, as required. However, the plan cannot be rebound because it is in use.

You can use the DB2 UET ISPF interface to find and cancel the threads that are running under that plan:

1. Display the Thread Filter Criteria panel (Option **1** on the Main Menu).
2. Specify the plan name as the filtering criteria, and press **F3**.
3. Open the Thread Summary Report panel. The panel lists only the threads that are running under the specified plan.
4. To select all of these threads for cancelation, type **C *** at the command line and press **Enter**. A "C" is displayed next to each thread.
5. Press **Enter** to issue the cancel command.
6. To determine whether the threads have actually terminated, type **REFRESH** at the command line and press **Enter**. Threads that have terminated no longer appear in the list.
7. If some threads do not terminate in a timely manner, you can use the **K** (Escalated Cancel) command to terminate the batch job, TSO user, or started task from which the threads originate.

After all threads that were running under the plan have terminated, you can rebind the plan.

Escalated thread cancelations

In DB2 UET, the DB2 -CANCEL THREAD command is usually used to cancel threads. However, in some circumstances, this command might not actually terminate a thread. To accommodate this situation, DB2 UET provides an alternative *escalated cancelation* method.

When you perform an escalated cancelation, DB2 UET issues the z/OS Cancel command to terminate the batch job, TSO user, or started task that is associated with the thread. DB2 UET can perform an escalated cancelation only for the following types of connections:

- A TSO user that has a TSO ATTACH request, the call attachment facility (CAF), or the Recoverable Resource Manager Services attachment facility (RRSAF) connection
- A batch job that has a TSO ATTACH request, CAF, or RRSAF connection
- An IMS™ DL/I batch job that has a IMSDLIBT connection

In a DB2 data sharing environment, escalated cancelations are limited to the DB2 subsystem that you selected from the interface that you are using. This subsystem is the one that you specified in a <DB2SYSTEM> element within the DSNUTILB intercept policy, in the DB2SSID parameter of a batch job, or on the Set DB2 System panel of the ISPF interface.

Escalated cancelation processing varies slightly for the different DB2 UET interfaces:

- If you use the DSNUTILB intercept or the batch interface, you enable escalated cancelation processing by setting the ESCALATE parameter to YES in the

*abpid*BCAN member for the DSNUTILB intercept (where *abpid* is the started task configuration ID) or in a CANCEL_THREADS request of a batch job step. When the utility or batch job runs, DB2 UET will issue the DB2 -CANCEL THREAD command first. Only if this command fails to terminate the thread will DB2 UET issue the z/OS Cancel command for an escalated cancelation.

- If you use the ISPF interface, you specify the `K` (Escalated Cancel) line command (instead of the `C` line command) for each thread for which you want to perform an escalated cancelation. DB2 UET will then issue the z/OS Cancel command directly, without first attempting to use the DB2 -CANCEL THREAD command.

Important: For an escalated cancelation to be performed from any DB2 UET interface, you must set the CANCEL_ESCALATION_ACTIVE option to YES in the started task initialization options member.

Thread filtering

In all DB2 UET interfaces, you can specify several types of thread-filtering criteria to identify the threads to cancel. The types of filtering criteria and the requirements for specifying them vary slightly by user interface.

Filtering criteria supported by each interface

From the ISPF and batch interfaces, you can specify the following types of filtering criteria:

- The thread's level of access to DB2 objects: Read (threads with lock states that allow read access only), Update (threads with lock states that allow update access only), or All (all threads including those without locks)
- The DB2 objects that the thread is referencing, including databases, table spaces, index spaces, partitions, tables, indexes, views, aliases, and synonyms
- The thread's correlation ID
- The connection ID and type
- The job name, program name, and address space identifier (ASID) from which the thread originated
- The current authorization identifier (authid) of the process that is associated with the active thread and the original authid of the process for which the thread was initially established
- The DB2 plan, package, and collection ID under which the thread is running
- Distributed Data Facility (DDF) requestor characteristics, including the IP address, the IP port number, and the location of a remote requestor
- The thread token value (batch interface only)

If you use the DSNUTILB intercept to block and cancel threads for the DB2 utilities, you can specify the following filtering criteria:

- Attributes of the DB2 utility for which you want to block and cancel threads, including the utility command, job name, user ID, and SHRLEVEL setting
- The DB2 objects that the utility needs to access, including databases, table spaces, index spaces, partitions, tables, indexes, views, aliases, and synonyms

Requirements for filtering criteria

The DB2 UET interfaces have different requirements for specifying thread-filtering criteria.

- In the ISPF interface, the use of thread-filtering criteria is optional. Any criteria that you specify on the Specify Thread Filter Criteria panel are used only to limit the list of active threads so that you can more easily find the ones that you want to view or cancel. If you specify no filtering criteria, all active threads are listed.
- In a batch thread-cancelation job step, you must specify the THREAD_TOKEN parameter or at least one of the other thread-filtering parameters for each CANCEL_THREADS request. If you do not specify one of these parameters for a cancel request, no threads will be canceled for that request. To perform thread blocking for a cancel request, you must specify at least one parameter for a DB2 object (for example, a table space).
- If you are using the DSNUTILB intercept, you must define at least one inclusion rule in the intercept policy that matches actual data for your DB utilities or objects. To define rules for filtering criteria in the intercept policy, use one or more <RULE> elements and the appropriate container elements (<RULESET>, <EXCLUDE>, and <INCLUDE>). If you specify no inclusion rules in your intercept policy, no intercept processing will occur.

Backout processing

By default, backout processing is enabled for normal thread cancelations that use the DB2 -CANCEL THREAD command and for escalated cancelations. If you use the DSNUTILB intercept or the batch interface to cancel threads, you can circumvent backout processing, if necessary.

When backout processing is enabled, if an outstanding unit of recovery exists for a thread that you are canceling, the uncommitted changes in that unit of recovery are backed out (rolled back) to prevent data inconsistencies in your DB2 tables. Usually, this behavior is preferred.

If you need to circumvent backout processing, you can specify one of the following options for the CANCEL_TYPE parameter in the *abpid*BCAN member for the DSNUTILB intercept (where *abpid* is the started task configuration ID) or in a CANCEL_THREADS request within a batch job step:

- NOBACKOUT: This option checks for outstanding units of recovery prior to cancelation.
  - If DB2 UET finds an outstanding unit of recovery for a thread, the thread is not canceled and the cancel request terminates.
  - If DB2 UET does *not* find an outstanding unit of recovery, the thread is canceled and no backout processing occurs.
- NO_UR_CHECKING: This option cancels threads without checking for outstanding units of recovery or performing any backout processing. (This option is equivalent to the DB2 NOBACKOUT option.)

Because these options can result in data integrity problems in your database, you should avoid using them. However, in some unusual circumstances, they might be useful. For example, assume that you have a long-running application that is tying up DB2 objects and the related UR has many uncommitted changes. In this case, you might want to use the NO_UR_CHECKING option for the thread cancelation because recovering the updated DB2 objects to an earlier point in time will be faster than backing out all of the changes.

Pre- and post-cancel user exits

If you need to perform any additional processing prior to or after thread cancelations, you can optionally create a pre-cancel user exit, post-cancel user exit, or both. These user exits will be called for the thread cancelations that you perform from any DB2 UET interface.

You must write the user exits in assembler language. DB2 UET provides sample user exits and DSECTS in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specify during product customization.

Using the sample exits as a starting point, you can define exits that meet your site-specific needs. For example, you can define a pre-cancel exit to prevent canceling threads that are running under certain plans, and define a post-cancel exit to obtain information for issuing cancelation notifications.

For DB2 UET to use your pre-cancel and post-cancel exits, you must specify the exit names in the started task initialization options member by using the PRE_CANCEL_EXIT and POST_CANCEL_EXIT options.

For more information about these exits, see "Pre-cancel exit" on page 456 and "Post-cancel exit" on page 462.

Related reference:

"Post-cancel exit" on page 462
You can create an optional post-cancel user exit to perform any processing that you consider to be necessary after canceling DB2 threads from the DB2 UET ISPF or batch interface. This exit applies to a single DB2 UET configuration.

"Pre-cancel exit" on page 456
You can create an optional pre-cancel user exit to perform any processing that you consider to be necessary prior to canceling DB2 threads from theDB2 UET ISPF or batch interface. The exit will apply to a single DB2 UET configuration.

Blocking and canceling threads by using the DSNUTILB intercept

You can use the DSNUTILB intercept to transparently block and cancel threads for the DB2 online utilities. The intercept can block and cancel threads for all DB2 utilities for which this function is appropriate.

For example, if you perform routine reorganizations of specific table spaces by using the REORG TABLESPACE utility, you could use the DSNUTILB intercept to automatically block and cancel threads on those table spaces whenever the utility is run by DSNUTILB.

To use the DSNUTILB intercept for this purpose, you must create an intercept policy that specifies one or more DB2 subsystems and at least one inclusion rule that selects the DB2 utilities, users, or objects for which to block and cancel threads. Use the XML elements that are valid for an intercept policy. For more information, see "Defining and using a DSNUTILB intercept policy" on page 129.

Optionally, you can set control parameters for thread cancelations in the *abpid*BCAN and *abpid*BGLB members (where *abpid* is the configuration ID that you specified during customization) in the *hlq.mlq*.SABPSAMP library. The DSNUTILB intercept will call the batch interface with these options to perform thread blocking and cancelation for the intercepted utilities.

Also, if you define an optional pre- or post-cancel exit, the intercept will call your exit for each worklist step (utility command) for which threads are to be blocked and canceled.

Related concepts:

Chapter 5, "Blocking and canceling DB2 threads," on page 173
You can cancel active DB2 threads by using any of the DB2 UET interfaces. By canceling threads, you can free DB2 objects that are being held so that other applications or utilities can access them.

"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

"Canceling threads by using the ISPF interface" on page 185
The DB2 UET ISPF interface provides a central point from which you can interactively monitor and cancel DB2 threads that originate from various sources in your environment.

"Blocking and canceling threads by using the batch interface" on page 208
The DB2 UET batch interface enables you to create a batch job step that cancels threads on the DB2 objects that an application or utility needs to access during batch processing. Optionally, this job step can also block new threads from forming on the objects that a utility needs until after the utility completes.

Supported DB2 utilities

DB2 UET supports thread blocking and cancelation for all DB2 utilities for which threads are initiated. For some utilities, thread-cancelation restrictions apply.

The following list presents the DB2 utilities for which thread blocking and cancelation is supported, with any applicable restrictions:

- CHECK DATA
- CHECK INDEX
- CHECK LOB
- COPY
- EXEC SQL ALTER
- EXEC SQL DROP
- LOAD
- QUIESCE
- REBUILD INDEX
- RECOVER
- REORG INDEX - If the SHRLEVEL CHANGE or SHRLEVEL REFERENCE option is specified for the REORG operation, active threads will be canceled but new threads will *not* be blocked.
- REORG TABLESPACE - If the SHRLEVEL CHANGE or SHRLEVEL REFERENCE option is specified for the REORG operation, active threads will be canceled but new threads will *not* be blocked.

You specify the utility for which to perform thread blocking and cancelation in the DSNUTILB intercept policy by using the <RULE> element with the UTILITY_COMMAND attribute.

Limiting duration of REORG utility jobs

An enhancement to the DB2 UET allows you to specify the DEADLINE option with REORG INDEX and REORG TABLESPACE if the SHRLEVEL option is omitted or if SHRLEVEL NONE is specified. Without this enhancement, the REORG INDEX and REORG TABLESPACE utilities do not allow you to use the DEADLINE option unless you specify SHRLEVEL REFERENCE or SHRLEVEL CHANGE.

When the DEADLINE option is encountered in a REORG INDEX or REORG TABLESPACE command, the utility syntax is manipulated as follows before invoking the DSNUTILB program:

- If the DEADLINE option is specified and SHRLEVEL NONE is defaulted, then SHRLEVEL REFERENCE is added to the REORG utility syntax.
- If the DEADLINE option is specified and SHRLEVEL NONE is coded, then SHRLEVEL REFERENCE replaces it in the REORG utility syntax.

For more information on the full syntax diagrams for REORG INDEX and REORG TABLESPACE, see *DB2 Version 9.1 for z/OS Utility Guide and Reference (SES1-2950-01)*. You need the DB2 UET to use this new capability in REORG INDEX and REORG TABLESPACE.

Considerations for blocking and canceling threads

Before you use the DSNUTILB intercept to block and cancel threads, consider these issues.

- DB2 UET will usually block new threads as well as cancel existing threads on the DB2 objects that a utility needs to access. However, thread blocking will not occur if the SHRLEVEL CHANGE option is specified for a utility or (for the REORG TABLESPACE utility only) if the SHRLEVEL REFERENCE option is specified.
- If the SHRLEVEL CHANGE option is specified for a utility, you should specify an EXCLUDE rule in the intercept policy that excludes the utility from thread-cancelation processing. Otherwise, DB2 UET will attempt to cancel threads instead of allowing other users to access objects during utility processing.
- To perform an escalated cancelation, you must set the ESCALATE parameter to YES in the *abpid*BCAN member and also set the CANCEL_ESCALATION_ACTIVE option to YES in the started task initialization options member.
- If you defined a pre-cancel exit or post-cancel exit and specified the exit name in the started task initialization options member, the exit will be called for each thread blocking and cancelation operation for a utility command (for each worklist step).
- If you specify the OPTIONS PREVIEW control statement in the input data set for a utility job step, DB2 UET will not block and cancel threads for any utility executions in preview mode. If you specify OPTIONS OFF to turn off the preview mode, thread blocking and cancelation can then occur.
- If you specify the PREVIEW option on the PARM in the JCL EXEC statement for a utility job step, DB2 UET will not block and cancel threads for any utility executions that are defined in the input data set for the job step. In this case, the OPTIONS OFF control statement has no effect.

Task flow for blocking and canceling threads

This task flow presents the high-level tasks that you will need to perform to block and cancel threads by using the DSNUTILB intercept.

The following table lists the tasks in the order in which they should be performed.

Table 25. Task flow for canceling threads by using the DSNUTILB intercept

Step	Task	Optional or required
1	Set the cancelation parameters in the *abpid*BCAN and *abpid*BGLB members of the SABPSAMP library (where *abpid* is the started task configuration ID), if you did not do so during customization. See "Customizing DSNUTILB intercept parameters (optional)" on page 93.	Optional (default values are available)
2	Create a DSNUTILB intercept policy. Use the <RULE>, <INCLUDE>, <EXCLUDE>, <RULESET>, and <USE_RULESET> elements, as appropriate, to select the DB utilities, users, or objects for which to block and cancel threads. See "Creating a DSNUTILB intercept policy" on page 160.	Required (otherwise DSNUTILB interception is excluded by default)
3	If you previously used the DSNUTILB intercept and temporarily deactivated it, you must reactivate the DSNUTILB intercept policy by using the appropriate z/OS console command. See "Reactivating the DSNUTILB intercept" on page 313.	Optional
4	After your DB2 utilities run, check the utility output and the started task output to ensure that threads were blocked and canceled as you intended. See "Determining whether thread blocking and canceling occurred" on page 184.	Recommended

Related concepts:

"Determining whether thread blocking and canceling occurred" on page 184
To determine whether thread blocking and cancelation processing occurred, you can check the DB2 UET messages in the SYSPRINT data set for the DB2 utility. Also, you can review the batch reports on threads canceled. Use SDSF or an equivalent tool to view this information.

Related tasks:

"Customizing DSNUTILB intercept parameters (optional)" on page 93
During customization with Tools Customizer, you set values for DSNUTILB intercept parameters on the Product Parameters panel with the required task **Create SABPSAMP members**. You can modify the cancel and global parameters that the DSNUTILB intercept uses for all thread blocking and cancelation activities.

"Creating a DSNUTILB intercept policy" on page 160
If you plan to use the DSNUTILB intercept to implement the enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads, you must define an intercept policy in XML. The policy specifies the DB2 subsystem or subsystems on which to perform intercept processing and optionally the rules for selecting the threads to block and cancel.

"Reactivating the DSNUTILB intercept" on page 313
By default, DSNUTILB interception is enabled globally across the z/OS image on which DB2 UET is running. However, if you deactivate the intercept, you will need to reactivate it to resume intercept processing.

Determining whether thread blocking and canceling occurred

To determine whether thread blocking and cancelation processing occurred, you can check the DB2 UET messages in the SYSPRINT data set for the DB2 utility. Also, you can review the batch reports on threads canceled. Use SDSF or an equivalent tool to view this information.

The following table lists key messages related to thread-cancelation processing. Look for these messages in the SYSPRINT data set for the DB2 utility.

Table 26. Key thread-cancelation messages in the utility SYSPRINT data set

Messages	Explanation
ABPU5006I *date time* Thread cancel started. Step=*step_number* ABPU5007I *date time* Thread cancel completed. RC=*return_code*	These messages indicate that thread-cancel processing for the specified worklist step began and then completed with the specified return code. The return code in ABPU5007I can be issued from either the DSNUTILB intercept or the batch interface.
ABPU5010I *date time* Allow threads started. Step=*step_number* ABPU5011I *date time* Allow threads completed. RC=*return_code*	These messages are issued if thread blocking occurred during the execution of the utility command (worklist step). The messages indicate that the allow-threads action (for allowing new threads to form again) began for the specified worklist step and then completed with the specified return code. The return code in ABPU5011I can be from either the DSNUTILB intercept or the batch interface. Note that no prior messages in the SYSPRINT data set for the utility indicate the start of thread blocking. However, additional messages regarding thread blocking are available in the SYSPRINT data set for the started task.
ABPU5301I *date time* Thread cancel prevented by policy	Some criteria in the DSNUTILB intercept policy prevented thread cancelation.

Note that the intercept calls the batch interface to perform thread blocking and cancelation. Therefore, some messages can contain return codes from the batch interface. Also, the batch reports on threads canceled and optionally on all active threads will be included in the SYSPRINT data set for the DB2 UET started task. You can review these reports to assess thread-cancelation activities for the intercepted utilities. To control whether the batch reports on active threads are generated, set the REPORT_TYPE parameter in the *abpid*BGLB member in the SABPSAMP library.

Tip: You should also look for the messages that apply to DSNUTILB intercept processing in general. These messages are available in the SYSPRINT data set for the utility and in the SYSPRINT data set for the started task. For more information, see "Determining whether DSNUTILB intercept processing occurred" on page 309.

Related concepts:

"Determining whether DSNUTILB intercept processing occurred" on page 309
You can check whether DSNUTILB intercept processing occurred as you expected for DB2 utilities by checking the DB2 UET messages that are incorporated into the SYSPRINT data set for the utility job and the SYSPRINT data set for the DB2 UET started task. Use SDSF or an equivalent tool to view this information.

"Task flow for blocking and canceling threads" on page 183
This task flow presents the high-level tasks that you will need to perform to block and cancel threads by using the DSNUTILB intercept.

Related reference:

"DB2 Utilities Enhancement Tool messages" on page 375
Look up DB2 UET messages to obtain information about them, including message explanations and suggested responses.

Canceling threads by using the ISPF interface

The DB2 UET ISPF interface provides a central point from which you can interactively monitor and cancel DB2 threads that originate from various sources in your environment.

From the ISPF interface, you can view summary or detailed information about active threads on an ad hoc basis to monitor thread activity or to research resource utilization or performance issues. If you determine that certain threads are tying up DB2 resources, you can cancel them.

The Thread Summary Report panel serves as the base panel from which to work with DB2 threads. By default, it displays all active threads on the DB2 subsystem that you selected on the Set DB2 System panel (including threads that originate from remote subsystems) and on any subsystems that are in the same DB2 data sharing group as the selected subsystem.

If your environment has a high level of DB2 activity, the Thread Summary Report panel might list a multitude of threads. To limit the number of threads that are displayed, you can create a thread filter. DB2 UET supports many types of filtering criteria, including the DB2 objects that the threads are referencing, the connection type, and the job or program from which the threads originate. For example, you can create a filter that specifies the DB2 objects that are having access problems so that you can view only the threads that are referencing these objects. On the Set Personal Defaults panel, if you set the **Save filter criteria** field to **Yes**, any filter criteria that you specify will apply to your future ISPF sessions, until you change these criteria.

After you find the threads that you are interested in, you can cancel them by specifying either the C (DB2 -CANCEL THREAD) or K (Escalated Cancel) command for each one. DB2 UET then issues the specified cancel command for the selected thread or threads.

After a cancel command is issued, you can check the thread status from the ISPF interface to determine if the thread is actually terminated. The Thread Summary Report panel displays the value `CAN` in the **Msg** column for any threads that you canceled with either the C or K command. After a thread is successfully terminated, it no longer appears in the list. If an error occurs during cancelation processing, the **Msg** column displays the value `ERR`. For threads with errors, you can view the error messages on the Cancel Thread Information panel.

Related concepts:

Chapter 5, “Blocking and canceling DB2 threads,” on page 173
You can cancel active DB2 threads by using any of the DB2 UET interfaces. By canceling threads, you can free DB2 objects that are being held so that other applications or utilities can access them.

“Blocking and canceling threads by using the DSNUTILB intercept” on page 180
You can use the DSNUTILB intercept to transparently block and cancel threads for the DB2 online utilities. The intercept can block and cancel threads for all DB2 utilities for which this function is appropriate.

“Blocking and canceling threads by using the batch interface” on page 208
The DB2 UET batch interface enables you to create a batch job step that cancels threads on the DB2 objects that an application or utility needs to access during batch processing. Optionally, this job step can also block new threads from forming on the objects that a utility needs until after the utility completes.

“Using the ISPF interface” on page 110
Learn about using the DB2 UET ISPF interface.

Thread Summary Report panel

The Thread Summary Report panel is the central point in the ISPF interface from which you can work with threads.

To access the panel, choose option **2** (Display and cancel threads) from the Main Menu. The panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   Thread Summary Report               15:19:13
Command ===>                                                 Scroll ===> PAGE

Commands: FILTER  REFRESH                                ABPID . . . : ABP1
                                                         DB2 system. : DBP2
Line commands: C - Cancel  S - Detail  O - Objects       User ID . . : PDABD
               K - Escalated Cancel    I - Info
                                                      Thread filter . : NO

                                                                    Row 1 of 4
 C   Msg Name     St A   Req ID           AUTHID  Plan     ASID Token  Member
 _       DB2CALL  T        2 DB2CALL      PDUSR1  PGMAPLAN 00CB    484
 _       DB2CALL  T        2 DB2CALL      PDUSR1  PGMAPLAN 00CB    485
 _       DB2CALL  T      196 DB2CALL      PDUSRZ  PGMBPLAN 00B8    454
 _       DB2CALL  T  *    43 DB2CALL      PDUSRX  PGMCPLAN 00AC    429
 _
```

Figure 21. Thread Summary Report panel

The panel lists all active threads on the selected DB2 subsystem (the subsystem for which the SSID is displayed in the **DB2 system** field), including threads that originate from remote subsystems. The panel also lists the threads on any subsystems that are in the same DB2 data sharing group as the selected subsystem. If you created a thread filter, the **Thread filter** field displays **YES** and the list contains only the threads that match *all* of your filtering criteria.

The panel supports several line commands and primary commands that you can use to work with the threads.

Thread Summary Report panel columns

The Thread Summary Report panel lists the following columns of information:

Msg Displays one of the following values that indicate the cancelation status:

- CAN if the thread was canceled with either the DB2 -CANCEL THREAD command or the z/OS Cancel command (that is, by a normal DB2 cancelation or an escalated cancelation)
- ERR if a cancel command was issued for the thread but cancelation processing ended with an error
- A blank if the thread is active and no cancel command has been issued for it

Name Displays the DB2 connection name that is supplied by the attachment facility and that is associated with a specific address-space connection.

St Displays the DB2 connection status code. Many codes are available and are described in the message DSNV404I in the IBM publication *DB2 for z/OS Messages*.

A Displays an asterisk (*) if the thread is active within DB2 or a blank if the thread is not active in DB2.

Req Displays the number of DB2 requests that were issued on the thread.

ID Displays the correlation ID of the thread. A correlation ID is an identifier that is associated with a specific thread. For example, it can be a TSO user ID or a job name.

AUTHID
Displays the current authorization identifier of the process that is associated with the thread. An authorization ID is a character string that can be verified for connection to DB2.

Plan Displays the name of the DB2 plan under which the thread is running.

ASID Displays the address-space identifier (in hexadecimal format) for the address space from which the thread originated.

Token Displays the thread token value for the thread. A thread token value is a numeric identifier that DB2 assigns to each active thread on a DB2 subsystem. A token value is unique to the DB2 subsystem or within a data sharing group to which the subsystem belongs. Threads that have a thread token value of zero are not displayed.

Member
If the thread is running on a DB2 subsystem in a data sharing group, this column displays the data sharing member name of the subsystem within the group.

Line commands

In the **C** column, you can type any of the following line commands next to a thread:

- C (Cancel) to perform a normal DB2 thread cancelation of thread by using the DB2 -CANCEL THREAD command
- S (Detail) to display detailed information about a thread
- O (Objects) to display the DB2 objects that a thread is referencing
- K (Escalated Cancel) to perform an escalated cancelation of a thread by using the z/OS Cancel command (available for certain connection types only)
- I (Info) to display the cancelation message log (available only for threads for which CAN or ERR is displayed in the **Msg** column)

You can specify these line commands next to multiple threads, if appropriate. For example, if you specify the S command next to two threads, the Display Thread Detail panel is displayed for the first selected thread in the list, and when you press F3, the panel is displayed again for the second selected thread. If you want to specify the same command (C, S, O, or K) next to all threads in the list, you can type the command at the command line followed by a space and then an asterisk (*). For example, type K * to select all threads for escalated cancelation processing. You can then deselect any threads that you do not want to process.

Also, you can specify different line commands next to different threads on the panel at the same time. For example, if you specify the S command next to two threads and specify the O command next to two different threads, the Display Thread Detail panel will be displayed for each thread for which the S command was specified and the Objects Referenced Report panel will be displayed for each thread for which the O command was specified.

Primary commands

At the command line, you can type either of the following primary commands:

- FILTER to change or define filtering criteria on the Thread Filter Criteria panel.
- REFRESH to update the data that is displayed for the listed threads by re-reading the data from the database. For example, after canceling a thread, you might want to issue the REFRESH command to check whether the thread has terminated. Note that the REFRESH operation does not affect any filter criteria that you applied to the thread list.

This panel also supports the standard CANCEL, FIND, HELP, and SORT commands. For descriptions of these commands, see "Primary commands" on page 118.

Task flow for canceling threads

Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

The following table lists the tasks in the general order in which they should be performed, although step 3 can be performed at any time.

Table 27. Task flow for canceling threads from the ISPF interface

Step	Task	Optional or required
1	Filter threads to find the ones that you want to cancel more easily. See "Filtering threads" on page 190.	Optional
2	Review the list of active threads to find the ones you want to cancel. See "Viewing a list of active threads" on page 189	Required
3	View detailed information about a thread or the DB2 objects that a thread is referencing to verify that it is one that you want to cancel. See "Viewing detailed information for a thread" on page 192.	Optional
4	Cancel threads by using either the C (Cancel) line command for a DB2 -CANCEL THREAD cancelation or the K (Escalated Cancel) line command for an escalated cancelation. See "Performing a normal DB2 cancelation of threads" on page 205 or "Performing an escalated cancelation of threads" on page 206.	Required

Table 27. Task flow for canceling threads from the ISPF interface (continued)

Step	Task	Optional or required
5	Review cancelation messages to determine if cancelation processing completed without errors. See "Determining whether threads were canceled" on page 207.	Recommended

Related tasks:

"Filtering threads" on page 190
You can create a filter to limit the threads that are displayed on the Thread Summary Report panel. This feature enables you to more easily find the threads that you want to research or cancel. DB2 UET "AND"s your filter criteria and displays only the threads that match *all* of the criteria.

"Viewing a list of active threads"
On the Thread Summary Report panel, you can view a list of the active threads that are referencing objects on the DB2 subsystem that you selected and on any active subsystem in the same data sharing group as the selected subsystem. You can then cancel a thread or drill down to detailed information about a thread.

"Viewing detailed information for a thread" on page 192
You can view detailed information about a thread that is listed on the Thread Summary Report panel to check its status or to verify that it is the one that you want to cancel.

"Viewing the DB2 objects that a thread is referencing" on page 193
You can view the DB2 objects that a thread is referencing to determine if the thread is holding any DB2 objects that are needed by an application or utility. You can then cancel the thread to free the objects, if necessary.

"Performing a normal DB2 cancelation of threads" on page 205
Usually, you cancel threads from the ISPF interface by specifying the `C` (Cancel) line command. This command issues the DB2 -CANCEL THREAD command to perform a normal DB2 thread cancelation. This type of cancelation cancels the selected threads only (not the jobs, TSO users, or started tasks from which they originated).

"Performing an escalated cancelation of threads" on page 206
If a normal DB2 cancelation of a thread fails to actually terminate the thread, you can specify the K (Escalated Cancel) line command to perform an escalated cancelation of the thread. An escalated cancelation issues the z/OS Cancel command to terminate the batch job, TSO user, or started task that is associated with the thread.

"Determining whether threads were canceled" on page 207
After canceling threads, you can view the messages that DB2 and DB2 UET generated for cancel processing to determine whether the threads were successfully canceled.

Viewing a list of active threads

On the Thread Summary Report panel, you can view a list of the active threads that are referencing objects on the DB2 subsystem that you selected and on any active subsystem in the same data sharing group as the selected subsystem. You can then cancel a thread or drill down to detailed information about a thread.

About this task

For each thread, the panel presents the summary thread attributes. This information is mostly the same information that is displayed when you issue the DB2 -DISPLAY THREAD command. From this information, you can determine

whether a thread has been canceled and assess its current status within DB2. This information might be useful when you researching database access problems.

Procedure

1. From the Main Menu, choose option **2** (Display and cancel threads). The Thread Summary Report panel is displayed. The panel lists the active threads in ascending order based on the **Name** column. If you previously created a thread filter, the list contains only the threads that match all of your filter criteria.

   Tip: If you cannot find the threads that you are interested in, you can use the FIND or SORT primary command to locate them, or use the FILTER primary command to define or edit a thread filter.
2. Review the columns of information for the threads that you are interested in. You can press the F11 key to scroll to the right, and press the F10 key to scroll to the left. To view detailed information about a specific thread or the DB2 objects that a thread is referencing, use the `S` (Detail) or `O` (Objects) line command.
3. When you are finished viewing thread information, press F3.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Filtering threads

You can create a filter to limit the threads that are displayed on the Thread Summary Report panel. This feature enables you to more easily find the threads that you want to research or cancel. DB2 UET "AND"s your filter criteria and displays only the threads that match *all* of the criteria.

About this task

When specifying filtering criteria, consider the following points:

- The input fields for filtering criteria are *not* case-sensitive.
- You can include a wildcard character in many of the filter fields. DB2 UET will match the wildcard pattern against information that DB2 maintains and display only the threads that match.
- To enter a long value in a scrollable field, you can press the F6 (Expand) key to display a pop-up window in which you can type the entire value, or you can scroll to the left and to the right in the field by using the F10 and F11 keys.

  Important: If a blank appears within values that you specify by using the Expand feature, apply IBM APAR OA10828 for ISPF version 4.0 to correct this problem.
- The filter fields correspond to attributes that DB2 maintains for DB2 objects and threads. For detailed information about these fields, refer to the documentation for your DB2 version.

To filter threads:

Procedure

1. Perform one of the following actions:
   - From the Main Menu, choose option **1** (Specify thread filter criteria).

- On the Thread Summary Report panel, type FILTER at the command line and press Enter.

The Thread Filter Criteria panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool        Thread Filter Criteria      15:52:25
 Command ===>

 Commands: RESET                                          ABPID . . . : ABP1
                                                          DB2 system. : DBP1
   Filter criteria will be used to limit the              User ID . . : PDABCD
   number of DB2 threads returned. Only those
   threads for which all of the following
   criteria are true will be displayed.
                                                                    More:     +
 ---------------------- Object Type and Access Filters ----------------------

   Access . . . . . ______    (All, Update, or Read)

   Database . . . . . ________

   Table space. . . . _________________
     Partition. . . . _____________________________________________________  >

   Index space. . . . _________________
     Partition. . . . _____________________________________________________  >

   Creator. . . . . _____________________________________________________
     Table. . . . . _____________________________________________________  >
   Creator. . . . . _____________________________________________________
     Synonym. . . . _____________________________________________________  >
   Creator. . . . . _____________________________________________________
     Alias. . . . . _____________________________________________________  >
   Creator. . . . . _____________________________________________________
     View . . . . . _____________________________________________________  >
   Creator. . . . . _____________________________________________________
     Index. . . . . _____________________________________________________  >

 ------------------------ Connection and Job Filters ------------------------

   Correlation ID . . ____________
   Connection ID. . . ________
   Connection type. . ________   (CAF, CICS, DBAT_DRD, DBAT_PP, IMSBMP, IMSCTL,
                                 IMSDLIBT, IMSMPP, IMSTBMP, RRSAF, TSO, or
                                 UTILITY)

   Auth ID. . . . . ________
   Original auth ID . ________
   Job name . . . . . ________
   ASID . . . . . ____       (Hex digits 0-9,A-F)

 --------------------- Package, Plan, and Program Filters ---------------------

   Collection ID. . .  _____________________________________________________  >
   Package name . . .  _____________________________________________________  >
   Program name . . .  _____________________________________________________  >
   Plan name. . . . .  ________

 ---------------------------------- DDF Filters ----------------------------------

   DDF location . . .  _____________________________________________________  >
   DDF IP address . .  ________________________________________________
   DDF IP port. . . .  _____
```

Figure 22. Thread Filter Criteria panel

2. Type a value in one or more of the following fields, as needed, to create a thread filter. Some fields require that other fields also be completed. Ensure that you complete all of the interrelated fields. Otherwise, the filtering criteria in these fields will not be applied. For more information about the panel fields, see "Thread Filter Criteria panel" on page 194.

3. Press F3 to return to the panel from which you started.

   Tip: If your filter criteria did not result in the expected list of threads, you can specify the **FILTER** primary command on the Thread Summary Report panel to return to the Thread Filter Criteria panel. You can then use the **RESET** primary command to clear all filtering criteria, or edit the filtering criteria.

Results

Your filter criteria will be saved from session to session if you specified **Yes** in the **Save filter criteria** field on the Set Personal Defaults panel when specifying your user settings. If you specified **No** instead, your filter criteria apply only to the current session.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Viewing detailed information for a thread

You can view detailed information about a thread that is listed on the Thread Summary Report panel to check its status or to verify that it is the one that you want to cancel.

About this task

For example, you might want to drill down to check whether a unit-of-recovery exists for the a thread before you cancel it.

Procedure

1. From the Main Menu, choose option **2** (Display and cancel threads). The Thread Summary Report panel is displayed.
2. Find the thread or threads for which you want to view detailed information. You can use the FIND, SORT, or FILTER primary commands to locate the threads of interest.
3. To select threads for which to display detailed information, as follows:
   - To select the threads individually, type the `S` (Detail) line command in the **C** column next to each one.
   - To select all threads in the list, type `S *` at the command line and press Enter. An **S** is displayed next to each thread.
4. Press Enter. The Display Thread Detail panel is displayed for the first thread that you selected, as follows:

```
DB2 Utilities Enhancement Tool   Display Thread Detail                18:53:42
Command ===>

Commands: INFO  REFRESH                                 ABPID . . . : ABP1
                                                        DB2 system. : DBP1
Cancel thread . . . NO    (Yes/No)                      User ID . . : PDABCD
Escalated cancel. . NO    (Yes/No)

------------------------------ Thread Detail -------------------------------
                                                                   More:     +
Thread token . . . . : 19              In-DB2 elapsed time. : 00:00:07.060944
Elapsed time . . . . : 00:00:07.06     In-DB2 CPU time. . . : 00:00:00.067937
Status . . . . . . . : T               Commit count . . . . : 1
Active . . . . . . . : In-DB2          Abort count. . . . . : 0
Auth ID. . . . . . . : PDUSER1         SQL calls. . . . . . : 362
Original auth ID . . : PDUSER1         GETPAGE count. . . . : 1790
Job name . . . . . . : PGMAJOB         Correlation ID . . . : PGMAPLAN
ASID . . . . . . . . : 012E            Connection ID. . . . : DB2CALL
UR start date. . . . : 12/03/2015      Connection type. . . : CAF
UR start time. . . . : 15:39:52.270    Log pages. . . . . . : 1
UR state . . . . . . :                 Log records. . . . . : 946
UR elapsed time. . . : 07:13:50.593
UR log start RBA . . : 000000000000B6F60C89
UR log end RBA . . . : 000000000000B6F60FA6

Plan name. . . . . . : PGMAPLAN
Member name. . . . . :
Program name . . . . : PROGRAMA                                          >
Package name . . . . : PROGRAMA                                          >
Collection ID. . . . : PROGRAMA                                          >
DDF location . . . . : RS123456                                          >

DDF IP port. . . . . :
DDF IP address . . . :
```

Figure 23. Display Thread Detail panel

5. Review the fields that are displayed for the thread. For more information about the fields, see "Display Thread Detail panel" on page 201.
6. When you are finished, press F3. If you selected multiple threads, the Display Thread Detail panel for the next selected thread is displayed. If you do not want to view details for any additional threads, type `CANCEL` at the command line and press Enter.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Viewing the DB2 objects that a thread is referencing

You can view the DB2 objects that a thread is referencing to determine if the thread is holding any DB2 objects that are needed by an application or utility. You can then cancel the thread to free the objects, if necessary.

About this task

Restriction: Claims without locks cannot be displayed on DB2 Version 8. Only claims with locks can be displayed on DB2 Version 8.

Procedure

1. From the Main Menu, choose option **2** (Display and cancel threads). The Thread Summary Report panel is displayed.
2. Select the thread or threads for which to display related DB2 objects, as follows:

- To select one or more threads individually, type the 0 (Objects) line command in the **C** column next to each one.
- To select all threads in the list, type 0 * at the command line and press Enter. An **O** is displayed next to each thread.

3. Press Enter. The Objects Referenced Report panel is displayed for the first or only thread that you selected, as follows:

```
DB2 Utilities Enhancement Tool   Objects Referenced Report            19:21:03
Command ===>                                                  Scroll ===> PAGE

Commands: INFO  REFRESH                                  ABPID . . . : ABP1
                                                         DB2 system. : DBP1
Cancel thread . . . NO    (Yes/No)                       User ID . . : PDABCD
Escalated cancel. . NO    (Yes/No)

Name     St A   Req ID           AUTHID   Plan     ASID Token Member
DB2CALL  T  *   359 DB2CALL      PDUSR1   PGMAPLAN  00C8 19

----------------------------- Objects Referenced -----------------------------

                                                                    Row 1 of 1
DB Name  PS Name  Part Lock Type    State Lock Count
DSNDB01  SCT02    0    PAGESET      IS    1
```

Figure 24. Objects Referenced Report panel

Tip: To ensure that you are viewing objects for the correct thread, check the thread attributes that are displayed in the list of objects referenced.

4. Review the columns of information about the referenced objects. For more information about a field, see "Objects Referenced Report panel" on page 204.
5. If you want to cancel the thread that is referencing the objects, specify **Yes** in the **Cancel thread** field to perform a standard DB2 cancelation, or specify **Yes** in both the **Cancel thread** field and the **Escalated cancel** field to perform an escalated cancelation. After canceling the thread, you can specify the **INFO** primary command to view the message log.
6. When you are finished, press F3. If you selected multiple threads, the Objects Referenced Report panel for the next selected thread is displayed. If you do not want to view information for any additional threads, type CANCEL at the command line and press Enter.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Panel fields reference

This reference section provides more information about the ISPF panels that you use to cancel threads.

- "Thread Filter Criteria panel"
- "Display Thread Detail panel" on page 201
- "Objects Referenced Report panel" on page 204

Thread Filter Criteria panel

Access

Specify one of the following options to indicate the level of access that the threads have to the DB2 objects that they are referencing:

- **All**: For all threads regardless of their access levels and lock states.

- **Update**: For the threads that have update access only. Update-access threads have one of the following object lock states: X (Exclusive), IX (Intent Exclusive), SIX (Share with Intent Exclusive), U (Update), or NSU (Non-share Update).
- **Read**: For the threads that have read access only. Read-access threads have an object lock state of S (Share) or IS (Intent Share).

For example, you might want to specify **All** if you need to rebind all plans that access a certain table after running the RUNSTATS utility and you do not care whether those plans currently have a lock on the table.

Note: The **Update** and **Read** options filter threads based on the types of locks on the specified DB2 objects. The **All** option filters threads based on information in the plan or package for accessing the specified objects. A plan or package can also identify objects that are related to the specified objects. In this case, DB2 UET displays threads for the both the specified objects and the related objects.

If you specify a value in this field, you must also specify a value in at least one of the following fields to identify the DB2 objects to which the access level applies: **Database**, **Table space**, **Index space**, **Partition**, **Table**, **Synonym**, **Alias**, **View**, **Index**, or any **Creator** (for any object for which you can specify A creator name).

Batch interface equivalent: ACCESS parameter.

Database

Type the name of a DB2 database that contains the objects that the threads are referencing. A database name can be up to eight characters long. Wildcards are permitted.

Batch interface equivalent: DATABASE parameter, or specified as part of the value for the TABLESPACE parameter or the INDEXSPACE parameter.

Table space

Type the name of a DB2 table space that contains the objects that the threads are referencing. A table space name can be up to eight characters long. Wildcards are permitted. To indicate a specific configuration of the table space, also specify a database name either in this field by using the format *database_name.tablespace_name* or in the **Database** field. If you do not specify a database name in either field, DB2 UET displays threads for all configurations of the named table space in all databases (as if only the % wildcard for zero or more characters is specified for the database name). If you want to select the threads that are referencing specific table space partitions, also specify a value in the corresponding **Partition** field.

Batch interface equivalent: TABLESPACE parameter.

Partition (table space)

Specify the table space partition or partitions that the threads are referencing by typing their partition numbers. Wildcards are *not* permitted.

A partition number can be an integer from 1 through 4096 for DB2 Version 8 or later. Specify multiple partitions as follows:

- A series of individual partitions: *integer1*,*integer2*,*integer3*
- A range of partitions: *integer1*:*integer3* (The first value must be less than the second value.)
- Both ranges and individual partitions: *integer1*,*integer2*:*integer4*,*integer5* (The first value must be less than the second value.)

When specifying multiple partition numbers, you can type up to 255 characters in this field. For your partition entry to be used in filtering threads, you must also specify a value in the **Table space** field, **Database** field, or both of these fields.

Batch interface equivalent: PART parameter.

Index space
Type the name of a DB2 index space that contains the index that the threads are referencing. An index space name can be up to eight characters long. Wildcards are permitted. To indicate a specific configuration of the index space, you can also specify a database name either in this field by using the format *database_name.indexspace_name* or in the **Database** field. If you do not specify a database name in either field, DB2 UET displays threads for all configurations of the named index space in all databases (as if only the % wildcard for zero or more characters is specified for the database name). If DB2 UET cannot detect any threads that hold locks on the index space, DB2 UET displays threads that hold locks on the table space that contains the table on which the index is created.

Batch interface equivalent: INDEXSPACE parameter.

Partition (index space)
Specify the index space partition or partitions that the threads are referencing by typing their partition numbers. Wildcards are *not* permitted.

A partition number can be an integer from 1 through 4096 for DB2 Version 8 or later. Specify multiple partitions as follows:

- A series of individual partitions: *integer1*,*integer2*,*integer3*
- A range of partitions: *integer1*:*integer3* (The first value must be less than the second value.)
- Both ranges and individual partitions: *integer1*,*integer2*:*integer4*,*integer5* (The first value must be less than the second value.)

When specifying multiple partition numbers, you can type up to 255 characters in this field.

For your entry to be used in filtering threads, you must also specify a value in the **Index space** field, the **Database** field, or both of these fields. To indicate partitions of a specific configuration of an index space or a table space, also specify a database name.

Batch interface equivalent: PART parameter.

Creator (table)
Type the creator ID for the tables that the threads are referencing. A creator name can be up to 128 characters long. Wildcards are permitted. You can also specify a value in the **Table** field to identify a specific table with this creator ID. If you do not specify a table, DB2 UET displays threads for all tables that were defined by this creator (as if only the wildcard for zero or more characters is specified in the **Table** field).

Batch interface equivalent: Specified as part of the value for the TABLE parameter.

Table Type the name of a DB2 table that the threads are referencing. A table name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted. You can also specify a value in the corresponding **Creator** field to identify

a specific table. If you do not specify a creator, DB2 UET displays threads for all tables that match the specified table name, regardless of their creators.

Batch interface equivalent: TABLE parameter.

Creator (synonym)
Type the creator ID for the synonyms for the tables that threads are referencing. A creator name can be up to 128 characters long. Wildcards are permitted. You can also specify a value in the **Synonym** field to identify a specific synonym with this creator ID. If you do not specify a synonym, DB2 UET displays threads for all synonyms that were defined by this creator (as if only the wildcard for zero or more characters is specified in the **Synonym** field).

Batch interface equivalent: Specified as part of the value for the SYNONYM parameter.

Synonym
Type the name of a DB2 synonym for the tables that the threads are referencing. A synonym name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted. You can also specify a value in the corresponding **Creator** field to identify a specific synonym. If you do not specify a creator, DB2 UET displays threads for all synonyms that match the specified synonym name, regardless of their creators.

Batch interface equivalent: SYNONYM parameter.

Creator (alias)
Type the creator ID for the aliases for the tables that threads are referencing. A creator name can be up to 128 characters long. Wildcards are permitted. You can also specify a value in the **Alias** field to identify a specific alias with this creator ID. If you do not specify an alias, DB2 UET displays threads for all aliases that were defined by this creator (as if only the wildcard for zero or more characters is specified in the **Alias** field).

Batch interface equivalent: Specified as part of the value for the ALIAS parameter.

Alias Type the name of a DB2 alias for the tables that the threads are referencing. An alias name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted. You can also specify a value in the corresponding **Creator** field to identify a specific alias. If you do not specify a creator, DB2 UET displays threads for all aliases that match the specified alias name, regardless of their creators.

Batch interface equivalent: ALIAS parameter.

Creator (view)
Type the creator ID for the views that are associated with the tables that the threads are referencing. A creator name can be up to 128 characters long. Wildcards are permitted. You can also specify a value in the **View** field to identify a specific view with this creator ID. If you do not specify a view, DB2 UET displays threads for all views that were defined by this creator (as if only the wildcard for zero or more characters is specified in the **View** field).

Batch interface equivalent: Specified as part of the value for the VIEW parameter.

View Type the name of a DB2 view that is associated with the tables that the threads are referencing. A view name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted. You can also specify a value in the corresponding **Creator** field to identify a specific view. If you do not specify a creator, DB2 UET displays threads for all views that match the specified view name, regardless of their creators.

Batch interface equivalent: VIEW parameter.

Creator (index)
Type the creator ID for the indexes that the threads are referencing. A creator name can be up to 128 characters long. Wildcards are permitted. You can also type an index name in the corresponding **Index** field to further qualify the index or indexes for which to display threads. If you do not specify an **Index** value, DB2 UET displays threads for all indexes that were defined by this creator (as if only the wildcard for zero or more characters is specified in the **Index** field).

Batch interface equivalent: Specified as part of the value for the INDEX parameter.

Index Type the name of a DB2 index that the threads are referencing. An index name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted. You can also specify a value in the corresponding **Creator** field to further qualify the index or indexes for which to display threads. If you do not specify a creator, DB2 UET displays threads for all indexes that match the specified index name, regardless of their creators. More specifically, DB2 UET displays the threads that hold locks on the index space that contains the index, or if no such threads are detected, DB2 UET displays the threads that hold locks on the table space that contains the table on which the index is created.

Batch interface equivalent: INDEX parameter.

Correlation ID
Type the correlation ID of a thread. A correlation ID is an identifier that is associated with a specific thread, for example, a TSO user ID or a job name. This ID can be up to 12 characters long. Wildcards are permitted.

Batch interface equivalent: CORRID parameter.

Connection ID
Type the connection ID that is supplied by the attachment facility and associated with the address space from which the threads originated. This ID can be up to eight characters in length.

Batch interface equivalent: CONNID parameter.

Connection type
Specify one of the following options to indicate the system region or program from which the thread originated or the type of distributed access protocols that the thread is using:

- **CAF**: The thread originates from a Call Attach Facility (CAF) user.
- **CICS**: The thread originates from a Customer Information Control System (CICS) region.
- **DBAT_DRD**: The thread is a distributed database access thread that uses distributed relational database access protocols (a DRDA® connection).

- **DBAT_PP**: The thread is a distributed database access thread that uses distributed database private protocols (a DDF connection).
- **IMSBMP**: The thread originates from an IMS Batch Message Processing (BMP) program.
- **IMSCTL**: The thread originates from an IMS control region.
- **IMSDLIBT**: The thread originates from an IMS DL/I batch region.
- **IMSMPP**: The thread originates from an IMS message processing program.
- **IMSTBMP**: The thread originates from an IMS transaction BMP.
- **RRSAF**: The thread originates from the Recoverable Resource Manager Service attachment facility.
- **TSO**: The thread originates from a TSO/E interactive or batch user.
- **UTILITY**: The thread originates from a DB2 utility.

For example, if you specify **CAF**, all threads from Call Attach Facility users will be displayed. You can also type the first part of an option name followed by a wildcard. For example, if you specify **DBAT%**, DB2 UET displays distributed database access threads with a connection type of DBAT_DRD or DBAT_PP.

Batch interface equivalent: CONNTYPE parameter.

Auth ID
Type the current authorization identifier of the process that is associated with the active threads. An authorization ID is a character string that represents an individual, an organizational group, or a function that can be verified for connection to DB2. This ID can be up to eight characters long. Wildcards are permitted. If you used the SET CURRENT SQLID statement to change the original authorization ID of the process, specify the ID that is currently used in this field and specify the ID that was initially used in the **Original auth ID** field.

Batch interface equivalent: AUTHID parameter.

Original auth ID
Type the original authorization identifier of the process for which the threads were initially established. This ID can be up to eight characters long.

Batch interface equivalent: ORIGINAL_AUTHID parameter.

Job name
Type the name of the job from which the threads originated. A job name is specified in the job card and can be up to eight characters long. Wildcards are permitted.

Batch interface equivalent: JOBNAME parameter.

ASID Type the address space identifier (in hexadecimal format) for the address space from which the threads originated. This value is a four-character hexadecimal string. Valid characters are the integers 0 through 9 and the letters A through F (or "a" through "f"). If you specify less than four characters, DB2 UET adds leading zeroes. For example, if you specify `9B`, DB2 UET converts this value to 009B. Wildcards are *not* permitted.

Batch interface equivalent: ASIDX parameter.

Collection ID
Type the collection identifier for the group of packages that includes the

package under which the threads are running. This ID can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters in later DB2 versions. Wildcards are permitted.

Batch interface equivalent: COLLID parameter.

Package name
Type the name of the DB2 package that the threads are running under. A package name can be up to 8 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted.

Batch interface equivalent: PACKAGE_NAME parameter.

Program name
Type the name of a program that is associated with the threads. A program name can be up to 8 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted. DB2 UET will display all threads running under a database request module (DBRM) that has a name matching the specified program name.

Batch interface equivalent: PROGRAM parameter.

Plan name
Type the name of the DB2 plan under which the threads are running. A plan name can be up to eight characters long. Wildcards are permitted.

Batch interface equivalent: PLANNAME parameter.

DDF location
Type the name of the Distributed Database Facility (DDF) requestor location from which the threads originated. A location name can be up to 16 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters in later DB2 versions. Wildcards are permitted. Any active database access threads that originated from the specified requestor location will be displayed.

Batch interface equivalent: DDF_DBAT_REQUESTOR_LOCATION parameter.

DDF IP address
Type the IP address of the DDF requestor from which the threads originated. An IP address is a unique address of a location on the Internet. If you have DB2 UDB for z/OS Version 7 or 8, this value must be an IPv4 address. An IPv4 address can be up to 15 characters long and must have the format *nnn.nnn.nnn.nnn*, where *nnn* is a number from 0 through 255. If you have DB2 Version 9.1 or later for z/OS, this value can be an IPv4 address, an IPv6 address, or a mixed-format IPv6-IPv4 address. An IPv6 address can be up to 39 characters long (including the delimiters) and must have the format *hhhh*:*hhhh*:*hhhh*:*hhhh*:*hhhh*:*hhhh*:*hhhh*:*hhhh* where *hhhh* is a four-digit hexadecimal value and a digit is any character from 0 through 9 and from A through F. A mixed-format address can be up to 45 characters long. Wildcards are *not* permitted in this field. This field overrides the connection ID, if specified. You can also specify a DDF IP port number to refine your filtering criteria.

Batch interface equivalent: DDF_DBAT_REQUESTOR_IPADDRESS parameter.

DDF IP port
Type the IP port number of the DDF requestor from which the threads originated. Valid port numbers are from 1 through 32767. Wildcards are

permitted. You must also specify the DDF IP address. DB2 UET can then display the active database access threads that originated from the remote IP address and port that you specified. These fields override the connection ID, if specified.

Batch interface equivalent: DDF_DBAT_REQUESTOR_IPPORT parameter.

Display Thread Detail panel

Thread token
Displays the thread token value for the thread. A thread token value is a numeric identifier that DB2 assigns to each active thread on a DB2 subsystem. A token value is unique to a subsystem or within a data sharing group. DB2 UET does not display threads that have a thread token value of zero.

Elapsed time
Displays the amount of time that elapsed between thread creation or sign on to DB2 and the display of the Display Thread Detail panel. The time is in the format HH:MM:SS.xx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xx* is hundredths of a second.

Status Displays the DB2 connection status code for the thread. Many codes are available and are described in the message DSNV404I in the IBM publication *DB2 for z/OS Messages*.

Active Displays one of the following values to indicate the thread status:

- In-DB2: The thread is active and running in DB2.
- In-APPL: The thread is not active in DB2.
- DEF-TERM: The thread is active and in deferred termination processing.
- Q-DEFTERM: The thread is active and is queued for deferred termination processing.

Auth ID
Displays the current authorization ID of the process that is associated with the thread. An authorization ID is a character string that can be verified for connection to DB2. If you used the SET CURRENT SQLID statement to change the original authorization ID of the process, this field displays the ID that is currently used and the **Original auth ID** field displays the ID that was initially used.

Original auth ID
Displays the original authorization identifier of the process for which the thread was initially established.

Job name
Displays the name of the job from which the thread originated; that is, the name that is specified in the job card. This field is blank if the thread is on a DB2 subsystem other than the one to which you are connected.

ASID Displays the address space identifier (in hexadecimal format) for the address space from which the thread originated. This field is blank if the thread is on a DB2 subsystem other than the one to which you are connected.

UR start date
Displays the start date (or creation date) of the unit of recovery (UR). A UR exists only if the thread modified DB2 objects or resources. The date is in the format MM/DD/YYYY, where *MM* is the month, *DD* is the day, and

YYYY is the year. If no UR exists or if no UR information is available (because the thread is on a data sharing member on another z/OS image), this field is blank.

UR start time
Displays the start time (or creation time) of the unit of recovery (UR). This time is in the format HH:MM:SS.xxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxx* is milliseconds. If no UR exists or if no UR information is available (because the thread is on a data sharing member on another z/OS image), this field is blank.

UR state
Displays one of the following values to indicate the status of the unit of recovery (UR):

- INFLIGHT: The UR is an active, inflight unit of work.
- IN PHAS1: The UR is currently in phase 1 of commit processing.
- IN PHAS2: The UR is currently in phase 2 of commit processing.
- COMMITED: Commit processing has ended.
- IN ABORT: The UR is being aborted. Rollback processing has begun but has not yet completed.
- ABORTED: The UR has been aborted. Rollback processing is complete.
- INDOUBT: The UR is in the in doubt state.

If no UR exists or if no UR information is available (because the thread is on a data sharing member on another z/OS image), this field is blank.

UR elapsed time
Displays the amount of time that elapsed between the start of the unit of recovery (UR) and the retrieval of this information for display. The time is in the format HH:MM:SS.xxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxx* is milliseconds. If no UR exists or if no UR information is available (because the thread is on a data sharing member on another z/OS image), this field is blank.

UR log start RBA
Displays the log start relative byte address (RBA) for the active unit of recovery (UR). If no UR exists or if no UR information is available (because the thread is on a data sharing member on another z/OS image), this field is blank.

UR log end RBA
Displays the log end RBA for the active unit of recovery (UR). This value is the RBA of the last log record written by the active UR. If no UR exists or if no UR information is available (because the thread is on a data sharing member on another z/OS image), this field is blank.

Plan name
Displays the name of the DB2 plan under which the thread is running.

Member name
If a thread is running on a DB2 subsystem that is a member of a data sharing group, displays the member name of that subsystem within the group.

Program name
Displays the name of the program that is currently running with the thread. If no SQL call activity occurred on the thread for a program, this field is blank.

Package name
Displays the name of the DB2 package under which the thread is running.

Collection ID
Displays the collection identifier for the group of packages that includes the package under which the thread is running.

DDF location
Displays the name of the Distributed Database Facility (DDF) requestor location from which the thread originated.

DDF IP port
Displays the IP port number of the DDF requestor from which the thread originated.

DDF IP address
Displays the IP address of the remote DDF requestor from which the thread originated. An IP address is a unique address of a location on the Internet.

In-DB2 elapsed time
Displays the amount of wall-clock time that elapsed while the thread was running in DB2. This interval is in the format HH:MM:SS.xxxxxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxxxxx* is microseconds.

In-DB2 CPU time
Displays the amount of CPU time that elapsed while the thread was running in DB2. This interval is in the format HH:MM:SS.xxxxxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxxxxx* is microseconds.

Commit count
Displays the number of DB2 commits that the thread performed.

Abort count
Displays the number of DB2 abort operations that the thread performed.

SQL calls
Displays the number of SQL calls that the thread performed.

GETPAGE count
Displays the number of DB2 getpages operations that were performed on behalf of the thread.

Correlation ID
Displays the correlation ID of the thread. A correlation ID is an identifier that is associated with a specific thread, for example, a TSO user ID or a job name.

Connection ID
Displays the connection ID that is supplied by the attachment facility and that is associated with the specific address space from which the thread originated.

Connection type
Displays the system region or program from which the thread originated or the type of distributed access protocols that the thread is using:

- CAF: The thread originates from a Call Attach Facility (CAF) user.
- CICS: The thread originates from a Customer Information Control System (CICS) region.
- DBAT_DRD: The thread is a distributed database access thread that uses distributed relational database access protocol (a DRDA connection).

- DBAT_PP: The thread is a distributed database access thread that uses distributed database private protocols (a DDF connection).
- IMSBMP: The thread originates from an IMS Batch Message Processing (BMP) program.
- IMSCTL: The thread originates from an IMS control region.
- IMSDLIBT: The thread originates from an IMS Data Language/I (DL/I) batch region.
- IMSMPP: The thread originates from an IMS message processing program.
- IMSTBMP: The thread originates from an IMS transaction BMP.
- RRSAF: The thread originates from the Recoverable Resource Manager Service attachment facility.
- TSO: The thread originates from a TSO/E interactive or batch user.
- UTILITY: The thread originates from a DB2 utility.

Log pages
Displays the number of physical log pages that contain information for the current unit of recovery. If no UR exists or if no UR information is available, this field displays a zero.

Log records
Displays the number of log records that were created for the active unit of recovery. Multiple log records can comprise a single log page. If no UR exists or if no UR information is available, this field displays a zero.

Objects Referenced Report panel

DB Name
Displays the name of the database that contains the referenced DB2 object.

PS Name
Displays the name of the page set that is referenced. This name is a table space name or an index space name.

Part Displays the identifier for the page set partition (a table space partition) that is referenced or that contains the referenced object.

Lock Type
Displays one of the following values to indicate the type of DB2 lock on a referenced object:

- PAGESET: A page set lock
- PART_SPL: Selective partition or partition page set lock
- PART_NSP: Non-selective partition page set lock
- DATA_PAGE: Data page lock
- ROW: Row level lock
- INDEX_PAGE: Index page lock
- MASS_DELETE: Mass delete lock
- DRAIN_CSREAD: Cursor stability read drain lock
- DRAIN_RRREAD: Repeatable read drain lock
- DRAIN_WRITE: Write drain lock

For more information about these lock types, refer to your DB2 documentation.

State Displays the state of the DB2 lock on a referenced object. This state can be one of the following values:

- S: Shared lock. The lock owner and any concurrent processes can read but not change data in the table space, partition, or table.
- U: Update lock. The lock owner can read but not change the locked data. To change the data, the lock owner can promote a U lock to an X lock. Concurrent processes can acquire S locks but not U locks. U locks are intended to reduce the chance of deadlocks.
- X: Exclusive lock. The lock owner can read or change data in the table space, partition, or table. Concurrent processes can access the data under limited conditions.
- SIX: Share with intent exclusive lock. The lock owner can read or change data in the table space, partition, or table. Concurrent processes can read data in the table space, partition, or table, but cannot change the data.
- IS: Intent share lock. The lock owner can read data in the table space, partition, or table, but cannot change the data. Concurrent processes can read and change the data.
- IX: Intent exclusive lock. The lock owner and concurrent processes can read and change data in the table space, partition, or table.
- NSU: Non-shared update lock.

  For more information about these lock types, see the *DB2 Version 9.1 for z/OS Administration Guide*.

Lock Count
Displays the number of DB2 locks held on a referenced object.

Performing a normal DB2 cancelation of threads

Usually, you cancel threads from the ISPF interface by specifying the `C` (Cancel) line command. This command issues the DB2 -CANCEL THREAD command to perform a normal DB2 thread cancelation. This type of cancelation cancels the selected threads only (not the jobs, TSO users, or started tasks from which they originated).

About this task

The following procedure describes how to perform a normal DB2 cancelation of one or more threads from Thread Summary Report panel. You can also cancel a single thread from the Display Thread Detail panel and the Objects Referenced Report panel by specifying **Yes** in the **Cancel thread** field and **No** in the **Escalated cancel** field.

Restriction: DB2 UET does not display or cancel threads that have a thread token value of zero from any panels.

Procedure

1. From the Main Menu, choose option **2** (Display and cancel threads). The Thread Summary Report panel is displayed.
2. In the list of threads, find the threads that you want to cancel. You can use the FIND, SORT, and FILTER primary commands to locate the threads.
3. Select the thread or threads to cancel, as follows:
   - To select threads individually, type the `C` (Cancel) line command in the **C** column next to each one.

- To select all threads in the list, type `C *` at the command line and press Enter. A **C** is displayed next to each thread.

4. Press Enter. DB2 UET issues the DB2 -CANCEL THREAD command for all of the selected threads. This command schedules the thread for termination in DB2. The **Msg** column displays the value `CAN` next to each thread for which the command was issued. If an error is returned, the **Msg** column displays `ERR` instead.

   Tip: You can specify the `I` (Info) line command next to any thread for which `CAN` or `ERR` is displayed to view the messages from cancelation processing.

5. To verify that the thread or threads were terminated in DB2, type `REFRESH` at the command line and press Enter. If a thread no longer appears in the list, it has been terminated. If a thread still appears in the list, termination processing has not yet completed or the -CANCEL THREAD command could not terminate the thread.

   Important: If you exit the Thread Summary Report panel, the list will no longer display `CAN` or `ERR` to indicate which threads were canceled. Also, you will no longer be able to view the messages from this cancelation attempt from the ISPF interface.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Performing an escalated cancelation of threads

If a normal DB2 cancelation of a thread fails to actually terminate the thread, you can specify the K (Escalated Cancel) line command to perform an escalated cancelation of the thread. An escalated cancelation issues the z/OS Cancel command to terminate the batch job, TSO user, or started task that is associated with the thread.

About this task

The following procedure describes how to perform an escalated cancelation of one or more threads from Thread Summary Report panel. You can also perform an escalated cancelation of a single thread from the Display Thread Detail panel and the Objects Referenced Report panel by specifying **Yes** in both the **Cancel thread** field and the **Escalated cancel** field.

Restrictions: You can perform an escalated cancelation only on threads that have one of the following connection types: CAF, IMSDLIBT, RRSAF, or TSO. Also, in a DB2 data sharing environment, escalated cancelations are limited to the DB2 subsystem to which you are connected (the subsystem that you selected on the Set DB2 System panel).

Procedure

1. From the Main Menu, choose option **2** (Display and cancel threads). The Thread Summary Report panel is displayed.
2. Find the thread or threads for which you need to perform an escalated cancelation.

   Tip: You can use the FIND, SORT, and FILTER primary commands to locate the threads.

3. Select the thread or threads to cancel, as follows:
   - To select one or more threads individually, type the K (Escalated Cancel) line command in the **C** column next to each one.
   - To select all threads in the list, type K * at the command line and press Enter. A **K** is displayed next to each thread.
4. Press Enter. DB2 UET issues the z/OS Cancel command for all of the selected threads. The **Msg** column displays the value CAN next to each thread for which the command was issued. If an error is returned, the **Msg** column displays ERR instead.

   Tip: You can specify the I (Info) line command next to a thread for which CAN or ERR is displayed to view messages from cancelation processing.
5. To verify that the thread was actually terminated, type REFRESH at the command line and press Enter. If the thread no longer appears in the list, it has been terminated.

   Important: If you exit the Thread Summary Report panel, the list of threads will no longer display CAN or ERR to indicate which threads were canceled. Also, you will no longer be able to view the messages from this cancelation attempt from the interface.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Determining whether threads were canceled

After canceling threads, you can view the messages that DB2 and DB2 UET generated for cancel processing to determine whether the threads were successfully canceled.

About this task

For example, if the Thread Summary Report panel displays ERR next to a canceled thread, you could review the error messages that were issued for that cancel operation to troubleshoot problems.

Procedure

1. On the Thread Summary Report panel, type I (Info) in the **C** column next a thread for which CAN or ERR is displayed in the **Msg** column. The Cancel Thread Information panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   Cancel Thread Information           10:48:39
Command ===>                                                  Scroll ===> PAGE

Press F11 or F10 to scroll the entire message text left or right.
                                                                  Row 1 of 2
Message Log
ABPS0220I 2015-12-04 10:48:35.23 TCB: 008D0BF8 Session: 24B0C070 - CANCEL THREAD requested
for thread token 212
ABPS0211I 2015-12-04 10:48:35.29 DSNV426I  !D8D DSNVCT THREAD '000212' HAS BEEN CANCELED
```

Figure 25. Cancel Thread Information panel

2. To view the full text of a message, place your cursor on the line for the message, and press F11 to scroll right or press F10 to scroll left.

3. When you are finished viewing messages, press F3. The Thread Summary Report panel is redisplayed.

Related concepts:

"Task flow for canceling threads" on page 188
Review this task flow to determine the high-level tasks that you will need to perform to cancel threads from the ISPF interface.

Blocking and canceling threads by using the batch interface

The DB2 UET batch interface enables you to create a batch job step that cancels threads on the DB2 objects that an application or utility needs to access during batch processing. Optionally, this job step can also block new threads from forming on the objects that a utility needs until after the utility completes.

By incorporating a thread-cancelation job step into your batch jobs, you can ensure that the applications and utilities that run during the batch window can access to the DB2 resources that they need. This feature can help reduce nighttime calls to your DBAs to address access problems.

Thread blocking is optional. You can implement thread blocking in your thread-cancelation job steps to ensure that your batch utilities and applications will have exclusive access to the DB2 objects that they need. If you block threads, you must create a second job step to allow new threads to access the DB2 objects once again, after your utilities complete.

In a batch job, the sequence of job steps is as follows:

1. The job step for canceling active threads and blocking new threads on selected DB2 objects
2. The job steps for the applications or utilities that need to access the DB2 objects
3. The job step for allowing new threads to form on the DB2 objects once again

When thread blocking is enabled, the first job step changes the statuses of the DB2 objects so that new threads cannot update these objects, and the last job step resets the object statuses to their original values.

In a single batch job, you can include several job steps for blocking and canceling threads with corresponding job steps for allowing threads. For example, if your batch utilities need to access different DB2 objects, you can create a set of thread-cancel and allow-thread job steps for each one; this way, the threads on the objects that a utility needs to access are blocked and canceled shortly before the utility runs and are allowed to reform after that utility completes.

The batch interface cancels threads by using the DB2 -CANCEL THREAD command. You can specify the optional ESCALATE parameter to perform an escalated cancelation. For an escalated cancelation, DB2 UET first issues the DB2 -CANCEL THREAD command. Only if that command fails to terminate a thread doesDB2 UET then issue the z/OS Cancel command to terminate the TSO user, started task, or batch job that is associated with the thread.

The batch interface supports all of the same thread-filtering criteria as the ISPF interface plus a parameter for filtering threads based on a thread token value. Also, the batch interface supports parameters for controlling how often and how many times DB2 UET checks the termination status of a thread for which a cancel command was issued. The job output will include messages that indicate the checking activity and termination status for each thread.

Related concepts:

Chapter 5, "Blocking and canceling DB2 threads," on page 173
You can cancel active DB2 threads by using any of the DB2 UET interfaces. By canceling threads, you can free DB2 objects that are being held so that other applications or utilities can access them.

"Blocking and canceling threads by using the DSNUTILB intercept" on page 180
You can use the DSNUTILB intercept to transparently block and cancel threads for the DB2 online utilities. The intercept can block and cancel threads for all DB2 utilities for which this function is appropriate.

"Canceling threads by using the ISPF interface" on page 185
The DB2 UET ISPF interface provides a central point from which you can interactively monitor and cancel DB2 threads that originate from various sources in your environment.

"Using the batch interface" on page 119
With the batch interface, you can manually create a job step for canceling threads and include it within a batch job.

Creating and running batch thread-cancelation job steps

Perform these high-level steps to create and run job steps for blocking and canceling threads and for allowing new threads to form again.

Before you begin

Before you begin coding a job step, review the syntax requirements for the basic JCL statements, the global parameters, and the CANCEL_THREADS parameters. Also review the example job steps.

About this task

To create and run batch thread-cancelation job steps:

Procedure

1. To create the job step for canceling active threads and optionally also for blocking new threads, perform these substeps:
   a. Write the required basic JCL statements. See "Basic JCL statements" on page 210.
   b. Add the global parameters. These parameters apply to the entire job step. See "Global parameters" on page 213.
   c. Add one or more CANCEL_THREADS commands. Each CANCEL_THREADS command must have at least one parameter that identifies the threads to cancel. See "CANCEL_THREADS command and parameters" on page 217.
2. If you specified the BLOCK_THREADS PARM on the EXEC statement in the thread-cancelation job step, you must create a separate job step that allows new threads to form once again. For this job step, you need to add only the basic JCL statements and a subset of the global parameters: ABPID, DB2SSID, and EXEC_TYPE (EXECUTE). For more information, see "Thread-blocker syntax requirements" on page 212.
3. Incorporate your batch job steps into a batch job, as appropriate.
4. Run the batch job in thread-cancelation simulation mode to check that the correct threads will be canceled. For instructions, see "Running a thread-cancelation job in a simulation mode" on page 233.

5. When you are satisfied with your batch job steps, run the batch job in execution mode.

What to do next

After the batch job runs, check the batch job output to determine if the thread-cancelation processing completed successfully.

Blocking threads

You can create a batch job step that both cancels active threads on the DB2 objects that your batch utilities or applications need to access and blocks new threads from updating these objects until after the utilities or applications complete.

By incorporating thread-blocking into your batch jobs, you can ensure that your batch utilities and applications will have exclusive access to the DB2 objects that they need. Any other utilities and applications that are running will not be able to create new threads on these DB2 objects from the time when existing threads are canceled until after your batch utility or application completes.

DB2 UET blocks new threads by changing the status of the DB2 objects to one of the following values:

RO The utilities and applications that are running in your environment have read-only access to the DB2 objects. None of them can update the objects in any way (add, change, or remove data).

UT Only the DB2 online utilities can access to the DB2 objects. No applications can access the objects to either read or update data.

DB2 UET selects one of these statuses for a DB2 object based on the ACCESS parameter value (ALL, READ, or UPDATE) that is specified for the batch thread-cancel job step. If the ACCESS value is ALL (the default value) or READ, DB2 UET changes the object status to UT. If the ACCESS value UPDATE, DB2 UET changes the object status to RO.

You must create a second DB2 UET job step to allow new threads to form on the DB2 objects once again. You run this job step after the thread-blocker job step and after the job step for the utility for which you are canceling threads. This "allow threads" job step restores the original statuses of the objects. It also removes all records for the uniquely identified thread-blocking operation from the DB2 table in which DB2 UET stores thread-blocker metadata. This table must be located on the DB2 subsystem on which threads were blocked or on a subsystem within the same DB2 data sharing group as that subsystem.

Tip: If a cancel request within a thread-blocker job step fails or if one or more of the canceled threads do not actually terminate, the new object statuses might remain in effect. You can control whether the new statuses remain in effect or the original statuses are reinstated by setting the optional ON_FAILURE (TERMINATE RESET_OBJECT_STATUS YES|NO) parameter in the thread-blocker job step.

Basic JCL statements

A batch job step for canceling threads or performing a thread-blocker action must contain certain basic JCL statements.

The following statements are required:

```
//JOBNAME JOB ,'MYNAME',CLASS=A,MSGCLASS=X
//*
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000
//STEPLIB  DD   DISP=SHR,DSN=ABP.INSTALL.SABPLOAD
//*
//ABPPARMS DD   *
 <input control cards>
/*
//
```

To implement a thread-blocker action, you must add a PARM to the EXEC statement as follows:

```
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000,
//         PARM='blocker_id,action'
```

Descriptions of JCL statements:

JOB The JOB statement, also called the job card, identifies the job to the z/OS system and supplies a job name that the system uses to refer to the job.

EXEC The EXEC statement invokes the DB2 UET program. You should specify `REGION=0M` to obtain as much memory as needed to run the program. Also, specify `DYNAMNBR=1000` to ensure that the maximum number of files that DB2 UET can dynamically allocate will be sufficient.

If you are implementing a thread-blocker action, add `PARM='blocker_id,action'`

Where:

- *blocker_id* is a user-defined identifier for the thread-blocker function in a batch job. DB2 UET uses this ID in internal processing and includes it in the job output that is displayed while and after the job runs. This ID can be up to 32 characters long and can include any alphanumeric or special characters other than a blank, comma (,), or single quotation mark ('). Also, the ID must be unique across all BLOCK_THREADS actions that are handled by a single DB2 UET started task configuration.
- *action* is one of the following thread-blocker operations:
  - BLOCK_THREADS prevents new threads from forming on the DB2 objects for which you are canceling threads. You must specify this action in the thread-cancelation job step that runs before the utility job step. If you specify this action, the thread-cancelation job step will cancel existing threads and prevent new threads from forming until after the ALLOW_THREADS job step completes.
  - ALLOW_THREADS ends thread-blocking and allows new threads to form on the DB2 objects once again. This action also deletes all data that is associated with a specific *blocker_id* from the DB2 table that stores thread-blocker data. A job step for which this action is specified runs after both the BLOCK_THREADS job step and the application or applications for which threads were blocked.
  - DELETE_BLOCKER_ID removes all data that is associated with a specific *blocker_id* from the DB2 table that stores thread-blocker data. Normally, you do not need to specify this action because the standard ALLOW_THREADS job step performs this function. However, if the ALLOW_THREADS job step fails, you can run a job step with the DELETE_BLOCKER_ID action to clean up the DB2 table.

STEPLIB DD

The STEPLIB DD statement identifies the DB2 UET load library.

ABPPARMS DD

The ABPPARMS DD statement provides the DB2 UET input control cards that are processed when a thread-cancelation job runs. You can specify these control cards instream or in a separate data set. To specify the control cards instream, specify either `ABPPARMS DD *` or `ABPPARMS DD DATA` followed by the control cards that you want to use. If you use a separate data set, specify the data set name.

Types of input control cards

In job steps for blocking and canceling threads, you must specify input control cards under the ABPPARMS DD. The input control cards specify global parameters that apply to the entire job step and individual cancel requests.

The input control cards can be categorized as follows:

- **Global parameters**: These parameters are specified once for the job step. They identify the DB2 subsystem on which you want to cancel threads, the DB2 UET configuration that you want to use, the mode in which you want to run the job (called *execution type*), whether a thread-cancelation job continues if an error occurs, the set of reports to generate, and the escape character for delimiting a wildcard character in a parameter value when that character is used as part of the data value and not as a wildcard. If you are using the thread-blocking feature, you can also specify a parameter for the time interval that DB2 UET waits between blocking new threads from forming and canceling the existing threads that applications are using.
- **CANCEL_THREADS commands and parameters**: A CANCEL_THREADS command (also called a *cancel request*) cancels the threads that meet all of the thread-filtering parameters that you specify for that command. You can specify multiple CANCEL_THREADS commands, each with its own set of thread-filtering criteria, in a thread-cancelation job. The CANCEL_THREADS commands do not apply to the thread-blocker ALLOW_THREADS and DELETE_BLOCKER_ID actions; if you specify CANCEL_THREADS commands in job steps for one of these actions, the commands will ignored.

You must first specify the global parameters under the ABPPARMS DD. Then specify any CANCEL_THREADS commands and parameters that you want to use.

None of the parameter values are case-sensitive. You can continue a long parameter value to the next line, if necessary, without using any special line-continuation character.

If you create an ABPPARMS data set instead of specifying parameters instream, you can use the record format RECFM=VB (variable length, blocked) with a record length that is large enough for all information to fit on one line.

Thread-blocker syntax requirements

DB2 UET batch job steps that include an optional thread-blocker action have specific requirements for the basic JCL, global parameters, and cancel parameters.

To implement thread blocking, you must create two job steps:

- A job step that cancels existing threads on selected DB2 objects and blocks new threads from forming on those objects
- A job step that allows new threads to access the objects once again

In each of these job steps, you must add a PARM to the EXEC statement in the basic JCL to specify the thread blocker action to perform. In the PARM, you can specify one of the following types of thread-blocker actions:

- BLOCK_THREADS to prevent new threads from forming on the DB2 objects for which you are canceling threads.
- ALLOW_THREADS to end thread-blocking and allow new threads to form on the DB2 objects once again. This action also deletes all data that is associated with a specific *blocker_id* from the DB2 table that stores thread-blocker data.
- DELETE_BLOCKER_ID to remove all data that is associated with a specific *blocker_id* from the relevant DB2 table. Normally, you do not need to specify this action because the standard ALLOW_THREADS job step performs this function unless a failure occurs.

Also, you must ensure that the job steps contain the required global parameters and cancel parameters.

Related concepts:

"Global parameters"
In DB2 UET job steps for canceling threads or performing a thread-blocker action, you must specify one or more global parameters under the ADPPARMS DD. These parameters apply to the job step as a whole.

"CANCEL_THREADS command and parameters" on page 217
In DB2 UET job steps for canceling threads, you must specify one or more CANCEL_THREADS commands with parameters for selecting the threads to cancel.

Global parameters

In DB2 UET job steps for canceling threads or performing a thread-blocker action, you must specify one or more global parameters under the ADPPARMS DD. These parameters apply to the job step as a whole.

For thread-cancelation job steps, only the DB2SSID parameter is required. The other global parameters are optional.

For job steps that contain any type of thread-blocker action (BLOCK_THREADS, ALLOW_THREADS, and DELETE_BLOCKER_ID), all of the following parameters are required:

- ABPID *(product_configuration_id*)
- DB2SSID (*ssid* |)
- EXEC_TYPE (EXECUTE)

When the group attach name is used to customize an installation of DB2 UET instead of a specific DB2SSID member, the group attach name is passed to the SSID EXEC parameter in the VRUPDATE job.

If you set the EXEC_TYPE parameter to SIMULATE instead of EXECUTE, no thread blocker activity will be reported in the simulation output.

For thread-cancelation job steps that include the BLOCK_THREADS action, you can also specify the following optional global parameters:

- THREAD_QUIESCE_TIME (*seconds*) to define the time interval that DB2 UET waits between initiating thread blocking and canceling threads

- ON_FAILURE (TERMINATE RESET_OBJECT_STATUS YES|NO) to control whether DB2 UET restores the original statuses of DB2 objects for which threads are blocked if a CANCEL_THREADS request fails or if some of the canceled threads do not actually terminate

In all job steps, add the parameters immediately after the ABPPARMS DD line. You can list the parameters in any order. When specifying a parameter, type the parameter name, a space, and then a valid value or option, for example:

```
DB2SSID
```

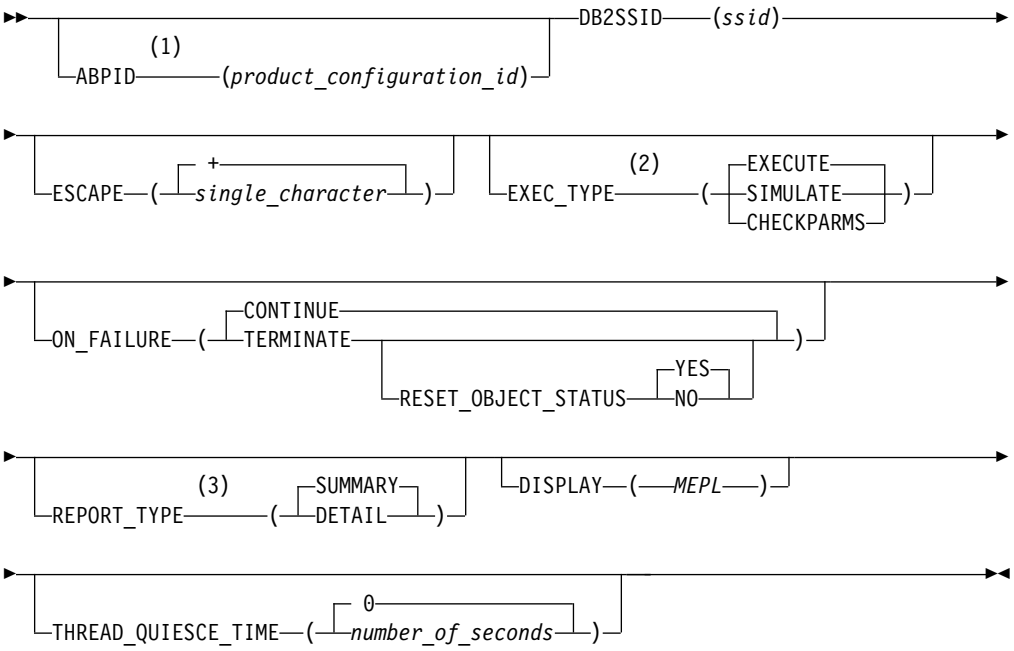
Related concepts:

"Thread-blocker syntax requirements" on page 212
DB2 UET batch job steps that include an optional thread-blocker action have specific requirements for the basic JCL, global parameters, and cancel parameters.

Syntax diagram for the global parameters

This syntax diagram illustrates the correct syntax to use for the global parameters in batch thread-cancelation or thread-blocker job steps.

DB2 UET global parameters



Notes:

1 The ABPID parameter is required for all thread-blocker actions.

2 The EXEC_TYPE parameter must be set to EXECUTE for all thread-blocker actions. If you specify SIMULATE instead, the job will run in simulation mode but the output will not include thread-blocker activities.

3 The REPORT_TYPE parameter is not supported for any thread-blocker actions.

Descriptions of global parameters

Refer to these global parameter descriptions when adding global parameters to your thread-cancelation and thread-blocker job steps. The descriptions indicate the purpose of a parameter, whether it is optional or required, and the default value (if provided).

A parameter is considered to be *optional* if it has a default value or does not need to be included in the job. A default value indicates what happens when the parameter is omitted from the control cards.

Parameters:

DB2SSID (*ssid*)

(Required) The ID of the active DB2 subsystem that you want to run DB2 UET against. This value can be up to four alphanumeric characters long. No default value is provided.

Tip: Ensure that the DB2 UET plan is bound on the subsystem that you specify. Otherwise, when the started task attempts to connect to the subsystem, the following SQL error will be issued:

```
Connection to DB2 failed. SSID=DBP1
 DSNT408I SQLCODE = -923, ERROR: CONNECTION NOT ESTABLISHED: DB2
          ACCESS, REASON 00E30301, TYPE 00000800, NAME ABCDPLAN
 DSNT418I SQLSTATE = 57015 SQLSTATE RETURN CODE
 DSNT415I SQLERRP = DSNAET03 SQL PROCEDURE DETECTING ERROR
Disconnection from DB2 was successful. SSID=DBP1
```

ESCAPE (*character*)

(Optional) The escape character that you want to use to delimit a wildcard character in a thread-filter parameter value when that character is used as an actual part of the data value and not as a wildcard. If you do not specify the escape character immediately before such a character, DB2 treats the character as a wildcard when performing pattern matching during thread filtering. (For more information about escape expressions and pattern matching, see the *DB2 Version 9.1 for z/OS SQL Reference*.) The default escape character is the plus sign (+). DB2 UET uses this default character if you do not specify a value, or if you specify one of the following characters: a blank symbol ␣, percent sign (%), or underscore (_).

Restriction: This parameter is ignored in environments that use a double-byte character set (DBCS).

EXEC_TYPE (CHECKPARMS | EXECUTE | SIMULATE)

(Optional) The mode in which the thread-cancelation job runs (also referred to as the *execution type*). The following options are valid:

- CHECKPARMS: Specify this option to have DB2 UET validate the syntax of all of the input parameters that are specified without actually running the job and canceling threads.
- EXECUTE: Specify this option to actually run a thread-cancelation job. All threads that match the thread-filtering parameters that you specify for the CANCEL_THREADS requests will be canceled. If you intend to use the thread-blocker feature, you must specify this option.
- SIMULATE: Specify this option to perform a trial run of a thread-cancelation job. DB2 UET simulates the cancelation of threads that match any filtering parameters you specify. The processing flow, messaging, and reporting are the same as if you actually ran the job;

however, no threads are canceled. You can use this option to check which threads will be canceled prior to running the job and to troubleshoot errors that might occur.

The default value is EXECUTE. You must use this default value for all thread-blocker actions. If you specify SIMULATE instead, no thread-blocker activity will be reported in the simulation output.

ABPID (*configuration_identifier***)**
(Optional for thread cancelations, required for all thread-blocker actions) An identifier for a DB2 UET started task configuration. This ID must be four alphanumeric characters long. Use this parameter to connect to a specific DB2 UET started task configuration, which has the initialization options and policies that you want to use. If DB2 UET cannot locate the specified started task or the started task cannot connect to the DB2 subsystem that is specified in the DB2SSID parameter, the batch job will terminate. If you do not specify this parameter for a thread-cancelation job, DB2 UET will attempt to locate an active DB2 UET started task that can connect to the DB2 subsystem that you specify in the DB2SSID parameter.

ON_FAILURE (CONTINUE | TERMINATE [RESET_OBJECT_STATUS YES | NO])
(Optional) Whether a thread-cancelation job continues or terminates when an error occurs during cancelation processing. By default, when an error occurs, the job continues. However, if you specify TERMINATE for this parameter, the job will terminate and no additional CANCEL_THREADS requests will be processed.

If you are using the thread-blocker feature, you can add a subparameter on the TERMINATE option to specify whether DB2 UET reinstates the original statuses of the DB2 objects when a CANCEL_THREADS request fails or one or more of the canceled threads do not actually terminate. During thread blocking, DB2 UET changes the statuses of the DB2 objects to either RO (read only) or UT (only the DB2 utilities have access). You can control whether DB2 UET leaves the object statuses in this state after a cancel failure by adding the RESET_OBJECT_STATUS subparameter, as follows:

```
ON_FAILURE (TERMINATE RESET_OBJECT_STATUS YES | NO)
```

If you specify `YES` (the default value) for this subparameter, DB2 UET reinstates the original statuses of the objects. If you specify `NO`, DB2 UET retains the object statuses that were in effect during thread blocking. This subparameter applies only to thread-cancelation job steps that specify `PARM='blocker_id,BLOCK_THREADS'` in the EXEC statement.

REPORT_TYPE (SUMMARY | DETAIL)
(Optional) The level of reporting that DB2 UET provides during an actual or simulated run of a thread-cancelation job. DB2 UET can generate up to five reports for each CANCEL_THREADS request as part of the job output. The following options are valid:

- SUMMARY: Specify this option to print the Threads Canceled report and the Threads Canceled Unit of Recovery report for every cancel request in a thread-cancelation job for which active threads are selected for cancelation based on the thread-filtering parameters in the request.
- DETAIL: Specify this option to print the following five reports for each cancel request: the Threads Canceled report, the Threads Canceled Unit of Recovery report, the All Active Threads report, the All Active Threads Unit of Recovery report, and the All Active Threads Objects Referenced report. If no threads are selected for cancel processing for a cancel

request (based on the specified thread-filtering parameters), DB2 UET generates only the three reports on all active threads.

The default value is SUMMARY. This parameter does not apply thread-blocker actions.

DISPLAY (MEPL)
(Optional) Enables the user to print a report of all DB2 UET modules that are allocated to the started task whose ABPID is specified in the parameter ABPID. For each module, the list shows the module maintenance level, the date and time when the module was built, and APAR level applied to each module for diagnostic use. Usually, this diagnostic information is requested by customer support.

THREAD_QUIESCE_TIME (*seconds*)
(Optional) If you are using the thread-blocker feature, you can specify the number of seconds that DB2 UET waits between initiating thread blocking and canceling the active threads that are identified by thread-filter parameters in a CANCEL_THREADS request. This interval is intended to allow the applications and utilities that are using the existing threads to complete units of work or quiesce prior to thread cancelation. If you do not specify this parameter, the default value of 0 is used, which causes DB2 UET to cancel existing threads immediately after initiating thread blocking. This parameter applies only to thread-cancelation job steps that specify PARM='*blocker_id*,BLOCK_THREADS' in the EXEC statement.

CANCEL_THREADS command and parameters

In DB2 UET job steps for canceling threads, you must specify one or more CANCEL_THREADS commands with parameters for selecting the threads to cancel.

In thread-cancelation job steps that specify the BLOCK_THREADS action, you must include at least one CANCEL_THREADS command with one or more of the following parameters for identifying the DB2 objects for which to block and cancel threads:

- ALIAS
- DATABASE
- INDEX
- INDEXSPACE
- SYNONYM
- TABLE
- TABLESPACE
- VIEW

If some of the cancel requests do not specify any DB2 objects, DB2 UET will issue a warning message with the return code RC=04 for each of these requests. The message and return code will indicate that new threads were not blocked for the request.

You do not need to include CANCEL_THREADS commands and parameters in ALLOW_THREADS or DELETE_BLOCKER_ID job steps. If you do so, the cancel specifications will be ignored.

Specify the CANCEL_THREADS commands after the global parameters. You can specify one or more of these commands. The number of CANCEL_THREADS commands is unlimited.

For each command, you must specify at least one parameter for selecting the threads to cancel. You can specify either:

- The THREAD_TOKEN parameter to cancel a thread that has a specific thread-token value
- One or more of the other thread-filtering parameters (for example, DATABASE and TABLESPACE) to cancel a subset of active threads that meet your filtering criteria

When specifying a CANCEL_THREADS command, specify `CANCEL_THREADS`, a space, and then all of the parameters for the cancel request enclosed in parentheses:

`CANCEL_THREADS (`*`cancelation_parameters`*`)`

An open parenthesis "(" must follow the CANCEL_THREADS command and precede the list of parameters. A closing parenthesis ")" must follow the last parameter value.

If you specify multiple thread-filtering parameters for a cancel request, DB2 UET "ANDs" these parameters so that only the threads that meet all of the thread-filtering criteria are canceled.

The following example contains three cancel requests. For the first cancel request, only the threads that match both the DATABASE and TABLESPACE criteria will be eligible for cancelation processing.

```
//ABPPARMS DD   *
  <global parameters>
  CANCEL_THREADS (
    DATABASE ABCDB%
    TABLESPACE XYZ
       )
  CANCEL_THREADS (
    PLANNAME ABCPL%
       )
  CANCEL_THREADS (
    DDF_DBAT_REQUESTOR_LOCATION VENUS
        )
 /*
 //
```

Also, you can specify any of the following optional cancelation control parameters anywhere in a CANCEL_THREADS request:

- CANCEL_TYPE to control how DB2 UET handles uncommitted work in any units of recovery for canceled threads (with or without backout processing)
- ESCALATE to control whether an escalated cancelation can be performed
- CHECK_THDTERM_RETRY_COUNT to control the maximum number of times that DB2 UET checks the termination status of threads during cancelation processing
- CHECK_THDTERM_RETRY_INTERVAL to control how frequently DB2 UET checks the termination status of threads during cancelation processing

Related concepts:

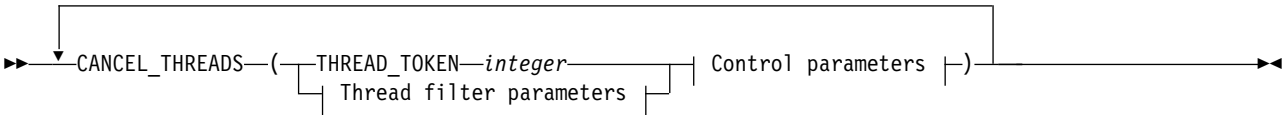
“Thread-blocker syntax requirements” on page 212
DB2 UET batch job steps that include an optional thread-blocker action have

specific requirements for the basic JCL, global parameters, and cancel parameters.

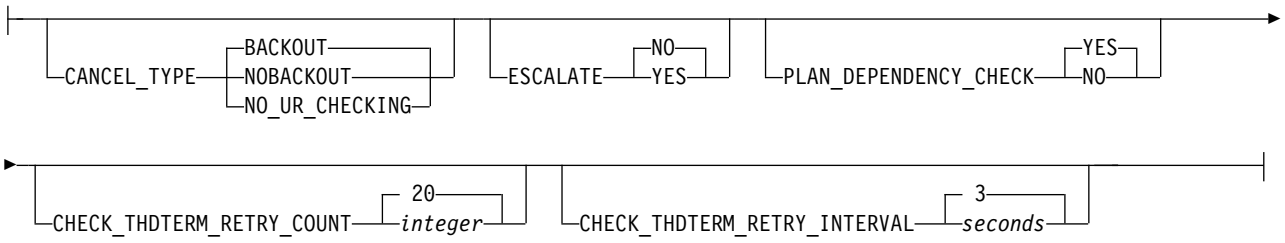
Syntax diagram for CANCEL_THREADS command and parameters

This syntax diagram illustrates the correct syntax to use for the CANCEL_THREADS command and parameters in batch thread-cancelation job steps.

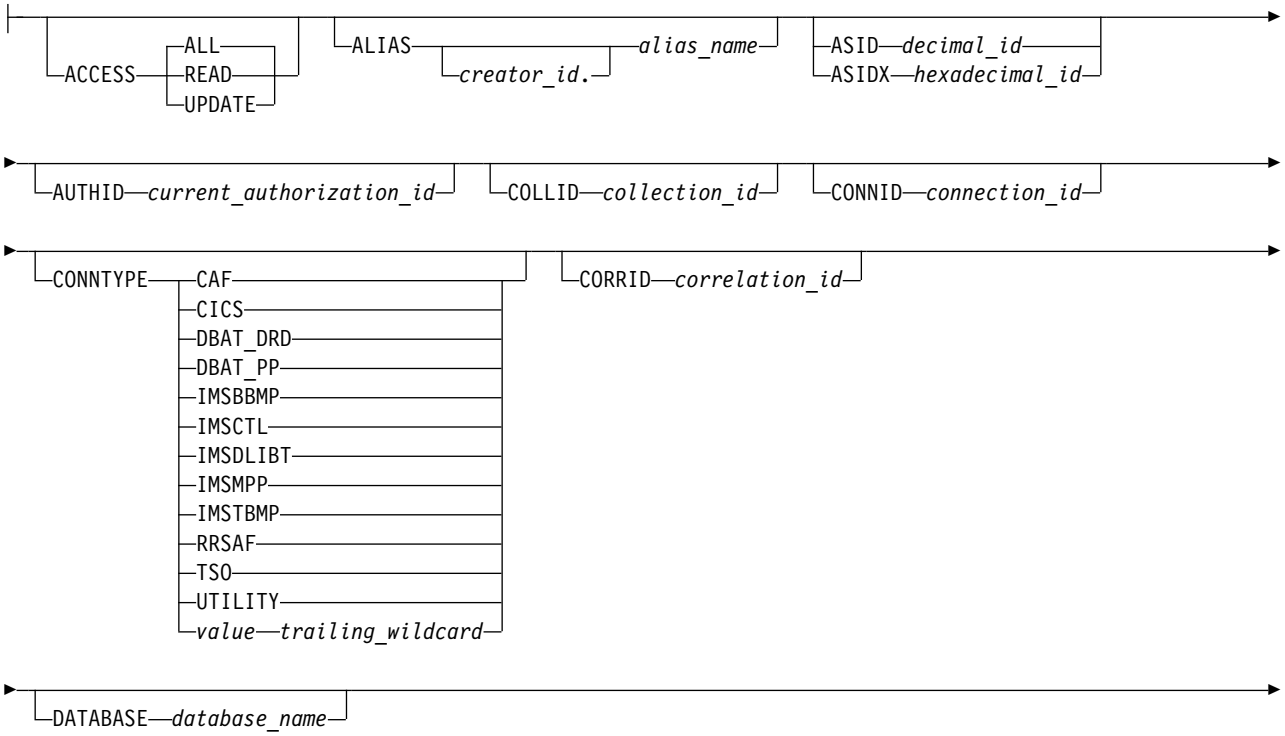
CANCEL_THREADS Command and Parameters

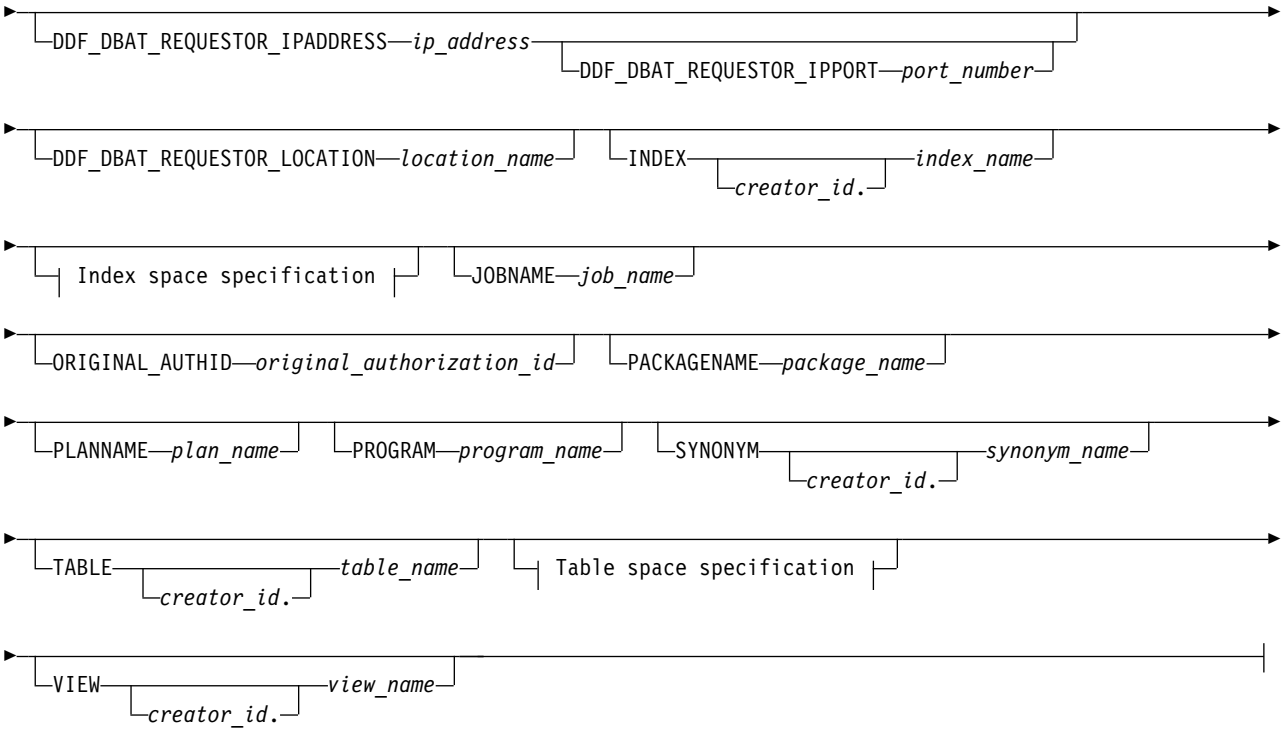


Control parameters:



Thread filter parameters:





Index space specification:

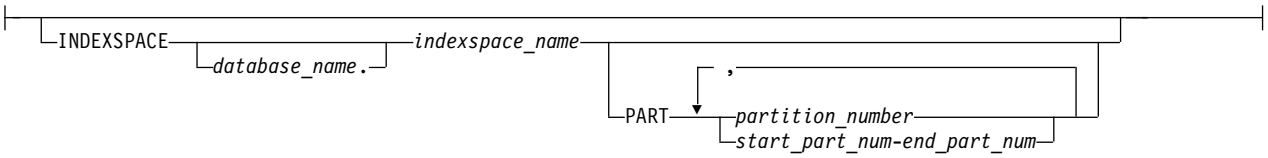
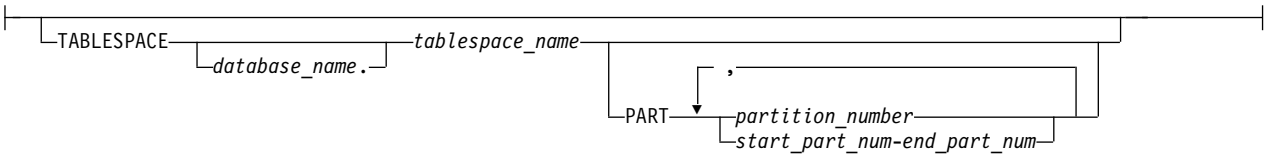


Table space specification:



Descriptions of cancel parameters

Refer to these CANCEL_THREADS parameter descriptions when adding the parameters to your thread-cancelation job steps. The descriptions indicate the purpose of a cancel parameter, whether it is optional or required, and the default value.

You must specify at least one parameter that identifies the threads to cancel: the THREAD_TOKEN parameter or one or more of the other thread-filtering parameters. The control parameters are optional.

The parameter descriptions include the default values, if available. A default value indicates what happens if you omit the parameter from a cancel request.

The parameter descriptions also indicate whether you can use the standard DB2 wildcards in a parameter value. If you specify a pattern of wildcards and literal values for a parameter, DB2 UET matches this pattern against the data that DB2 maintains for threads, objects, and connections to determine which threads to cancel.

Tip: If you do not know the value to enter for a specific thread-filter parameter such as THREAD_TOKEN, you can attempt to find the appropriate value by viewing thread attributes in one of the following ways:

- Create a batch thread-cancel job step that includes the threads that you are interested in. Specify the EXEC_TYPE (SIMULATE) and the REPORT_TYPE (DETAIL) parameters. Then run the job step in simulation mode. After the simulation run completes, review the reports for all active threads.
- If you have access to the DB2 UET ISPF interface, navigate to the Thread Summary Report panel. You can then drill down to the Display Thread Detail panel where you can view detailed information for a thread.
- Issue the DB2 **-DISPLAY THREADS** command.

Parameters:

ACCESS READ | UPDATE | ALL

Use this parameter to cancel threads based on their level of access to the DB2 objects that they are referencing. For this filter criterion to be applied, you must also specify at least one of the following parameters to identify the DB2 objects: DATABASE, TABLESPACE, PART, TABLE, ALIAS, SYNONYM, VIEW, or INDEX, with the corresponding creator ID if applicable.

The following ACCESS options are valid:

- READ: Specify this option to cancel threads that have read access only. Read-access threads have an object lock state of S (Share) or IS (Intent Share).
- UPDATE: Specify this option to cancel threads that have update access only. Update-access threads have one of the following object lock states: X (Exclusive), IX (Intent Exclusive), SIX (Share with Intent Exclusive), U (Update), or NSU (Non-share Update).
- ALL: Specify this option to cancel all threads regardless of their access levels and lock states. When this option is selected, DB2 UET cancels threads on the DB2 objects that you specified by using the object parameters and on any related DB2 objects that are identified in a plan or package for the specified objects. This option is the default value.

For example, you might use the default value of ALL if you need to rebind all plans that access a certain table after running the RUNSTATS utility and you do not care whether those plans currently have a lock on the table.

If you enable thread blocking for a cancelation job, DB2 UET will change the status of DB2 objects based on this ACCESS value. If the ACCESS value is READ or ALL, DB2 UET changes the object status to UT. If the ACCESS value is UPDATE, DB2 UET will change the status of the objects to RO.

ALIAS [*creator_id*.]*alias_name*

Use this parameter to cancel threads based on the DB2 aliases for the tables that the threads are referencing. You can specify either the alias name only or both the alias name and the creator ID for the alias. When specifying both the alias name and creator ID, use the format

creator_id.alias_name. An alias name can be 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. A creator name can be up to 128 characters long. Wildcards are permitted in both parts of the value. If you omit the creator ID, DB2 UET cancels threads for all aliases that match the specified alias name, regardless of their creator IDs.

If you enable thread blocking for the cancelation job, DB2 UET will change the status of the table space that contains the tables to which the specified alias refers to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

ASID *id*
Use this parameter to cancel threads based on the address space identifier (in decimal format) for the address space from which the threads originated. You can specify a decimal value from 1 through 65,535. Wildcards are *not* permitted.

If you specify this parameter, do not also specify the ASIDX parameter.

ASIDX *id*
Use this parameter to cancel threads based on the address space identifier (in hexadecimal format) for the address space from which the threads originated. This value is a four-character hexadecimal string. Valid characters are the integers 0 through 9 and the letters A through F (or "a" through "f"). If you specify less than four characters, DB2 UET adds leading zeroes. For example, if you specify `9B`, DB2 UET converts this value to 009B. Wildcards are *not* permitted.

If you specify this parameter, do not also specify the ASID parameter.

AUTHID *authorization_id*
Use this parameter to cancel threads based on the current authorization ID of the process that is associated with the active threads. An authorization ID is a character string that represents an individual, an organizational group, or a function and that can be verified for connection to DB2. The ID can be up to eight characters long. Wildcards are permitted. If you used the SET CURRENT SQLID statement to change the original authorization ID, specify the authorization ID that is currently used with this parameter and specify the authorization ID that was originally used with the ORIGINAL_AUTHID parameter.

CANCEL_TYPE BACKOUT | NOBACKOUT | NO_UR_CHECKING
Use this parameter to specify the type of cancel request. The following options are valid:

Important: The NOBACKOUT and NO_UR_CHECKING cancel types are not supported for threads on a DB2 subsystem other than one to which you are connected (the subsystem specified in the DB2SSID global parameter). In a DB2 data sharing environment, these options do not apply to other subsystems in the data sharing group.

- BACKOUT: Specify this option to have DB2 back out (roll back) any uncommitted work that exists in the current unit of recovery when a thread is canceled. This backout processing ensures that the data integrity of the updated DB2 objects is maintained. Any updates that occurred after the last commit operation are backed out.

- NOBACKOUT: **Do not specify this option if possible**. This option causes DB2 UET to check for any outstanding units of recovery just prior to canceling threads but prevents DB2 from performing backout processing. If DB2 UET finds an outstanding unit of recovery for a thread, the thread is *not* canceled and the return code RC=12 is issued. Depending on whether you set the ON_FAILURE parameter to TERMINATE, the CANCEL_THREAD request either terminates or continues to the next thread. By default, the request will continue and cancel the other threads. If DB2 UET does not find an outstanding unit of recovery, the thread is canceled. In this situation, data integrity problems can occur if an outstanding unit of recovery is created for a thread after DB2 UET has determined that no outstanding units of recovery exist for the thread and before the thread is canceled. Therefore, use this option with caution.
- NO_UR_CHECKING: **Do not specify this option unless necessary**. This option causes threads to be canceled without backout processing or any checking for outstanding units of recovery prior to cancelation. All threads that match your thread-filtering criteria are canceled, even if outstanding units of recovery exist for them. This situation is likely to result in data corruption in your database. (This option is equivalent to using the DB2 NOBACKOUT option on the -CANCEL THREAD command. For more information about the DB2 NOBACKOUT option, see the *DB2 for z/OS Command Reference*.)

The default option is BACKOUT.

CHECK_THDTERM_RETRY_COUNT ***integer***
After issuing the DB2 -CANCEL THREAD command, DB2 UET checks the status of the canceled threads to determine whether DB2 has actually terminated them. If some threads are not yet terminated, DB2 UET continues to check the thread status until all of the canceled threads are actually terminated or until the maximum retry count that you set with this parameter is reached. If the maximum retry count is reached and a thread has still not been terminated, DB2 UET issues an error message with the appropriate return code. If you also set the ESCALATE parameter to YES, DB2 UET will then issue the z/OS Cancel command to terminate the thread.

Valid values are from 1 through 32,767. The default value is 20.

Tip: You can also specify the optional CHECK_THDTERM_RETRY_INTERVAL parameter to define how frequently thread-status checking occurs.

CHECK_THDTERM_RETRY_INTERVAL ***seconds***
Use this parameter to specify how often (in seconds) DB2 UET checks the status of the threads that you canceled to determine whether they are actually terminated in DB2. Valid values are from 1 through 32,767. The default value is 3.

Tip: You can also specify the optional CHECK_THDTERM_RETRY_COUNT parameter to define the maximum number of times that DB2 UET will attempt thread-status checking.

COLLID ***collection_id***
Use this parameter to cancel threads based on the collection identifier for the group of packages that includes the package under which the threads

are running. This ID can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters in later DB2 versions. Wildcards are permitted.

CONNID *connection_id*
Use this parameter to cancel a thread based on the connection identifier that is supplied by the attachment facility and that is associated with the specific address space from which the threads originated. This ID can be up to eight characters long. Wildcards are permitted.

CONNTYPE *connection_type_option*
Use this parameter to cancel threads based on the system region or program from which the DB2 connection originated or the type of distributed access protocols that the connection is using. The following options are valid:

- CAF: The thread originates from a Call Attach Facility (CAF) user.
- CICS: The thread originates from a Customer Information Control System (CICS) region.
- DBAT_DRD: The thread is a distributed database access thread that uses distributed relational database access protocols (a DRDA connection).
- DBAT_PP: The thread is a distributed database access thread that uses distributed database private protocols (a DDF connection).
- IMSBMP: The thread originates from an IMS Batch Message Processing (BMP) program.
- IMSCTL: The thread originates from an IMS control region.
- IMSDLIBT: The thread originates from an IMS DL/I batch region.
- IMSMPP: The thread originates from an IMS message processing program.
- IMSTBMP: The thread originates from an IMS transaction BMP.
- RRSAF: The thread originates from the Recoverable Resource Manager Service attachment facility.
- TSO: The thread originates from a TSO/E interactive or batch user.
- UTILITY: The thread originates from a DB2 utility.

For example, if you specify `CONNTYPE CAF`, all threads from Call Attach Facility users would be terminated. You can also type the first part of an option name followed by a wildcard. For example, if you specify `CONNTYPE DBAT%`, DB2 UET terminates distributed database access threads from remote systems that have a connection type of DBAT_DRD or DBAT_PP.

CORRID *correlation_id*
Use this parameter to cancel threads based on their correlation IDs. A correlation ID is an identifier that is associated with a specific thread. For example, it can be either a TSO user ID or a job name. This value can be up to 12 characters long. Wildcards are permitted.

DATABASE *database_name*
Use this parameter to cancel threads based on the name of the database that contains the DB2 objects that the threads are referencing. A database name can be up to eight characters long. Wildcards are permitted.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of all table spaces in the specified database to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

Tip: You can also specify the optional TABLESPACE parameter to refine the set of threads to be terminated.

DDF_DBAT_REQUESTOR_IPADDRESS *ip_address*

Use this parameter to cancel active database access threads based on the IP address of the Distributed Database Facility (DDF) requestor from which they originated. An IP address is a unique address of a location on the Internet. If you have DB2 UDB for z/OS Version 7 or Version 8, this value must be an IPv4 address. An IPv4 address can be up to 15 characters long and must have the format *nnn.nnn.nnn.nnn*, where *nnn* is a number from 0 through 255. If you have DB2 Version 9.1 or later for z/OS or later, this value can be an IPv4 address, an IPv6 address, or a mixed-format IPv6-IPv4 address. An IPv6 address can be up to 39 characters long (including the delimiters) and must have the format *hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh* where *hhhh* is a four-digit hexadecimal value and a digit is any character from 0 through 9 and from A through F. A mixed-format address can be up to 45 characters long. Wildcards are *not* permitted.

This parameter overrides the CONNID parameter, if specified.

Tip: You can also specify the optional DDF_DBAT_REQUESTOR_IPPORT parameter to refine your thread-filtering criteria.

DDF_DBAT_REQUESTOR_IPPORT *port_number*

Use this parameter to cancel active database access threads based on the IP port number of the DDF requestor from which the threads originated. Valid port numbers are from 1 through 32,767. Wildcards are permitted.

You must also specify the DDF_DBAT_REQUESTOR_IPADDRESS parameter. DB2 UET will then cancel the active database access threads that originated from the remote IP address and port that you specify. These parameters override the CONNID parameter, if specified.

DDF_DBAT_REQUESTOR_LOCATION *location_name*

Use this parameter to cancel threads based on the name of the DDF requestor location from which they originated. A location name can be up to 16 characters long. Wildcards are permitted. Any database access threads that are currently active and that originated from the remote requestor location are canceled.

ESCALATE YES | NO

Use this parameter to specify whether DB2 UET issues the z/OS Cancel command to cancel a thread when the DB2 -CANCEL THREAD command fails to do so. If you set this parameter to YES, DB2 UET can issue the z/OS Cancel command to terminate the TSO user, batch job, or started task from which the thread originated. This escalated cancelation processing is supported only for the following connection types: CAF, IMSDLIBT, RRSAF, and TSO. Also, in a DB2 data sharing environment, escalated cancelation is supported only for threads on the subsystem that you specified with the DB2SSID parameter. If you set this parameter to NO, DB2 UET issues the DB2 -CANCEL THREAD command but does not escalate cancelation processing to the z/OS Cancel command when the DB2 -CANCEL THREAD command fails to terminate a thread; any threads that were not canceled remain active. The default value is NO.

Important: DB2 UET will not perform an escalated cancelation, even if you set the ESCALATE parameter to YES, in the following situations:

- A thread has a connection type that is not supported for escalated cancelation processing
- The CANCEL_ESCALATION_ACTIVE option in the started task initialization options file is set to NO

INDEX [*creator_id*.]*index_name*
Use this parameter to cancel threads that are referencing a DB2 index. You can specify either the index name only or both the index name and creator ID in the format *creator_id.index_name*. An index name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. A creator name can be up to 128 characters long. Wildcards are permitted in both parts of the value. If you omit the creator ID, DB2 UET cancels threads for all indexes that match the specified index name, regardless of their creator IDs. More specifically, DB2 UET cancels the threads that hold locks on the index space that contains the index, or if no such threads are detected, DB2 UET cancels threads that hold locks on the table space that contains the table on which the index is created.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the table space that contains the table on which the index is created to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

INDEXSPACE [*database_name*.]*index_space_name*
Use this parameter to cancel threads that are referencing a DB2 index space. You can specify the index space name only or both the index space name and a database name in the format *database_name.index_space_name*. An index space name and a database name can each be up to eight characters long. Wildcards are permitted in each part of the value. If you omit the database name or specify only the % wildcard for the database name in this parameter, and if the DATABASE parameter does not specify a database, DB2 UET cancels threads on all configurations of the specified index space in all databases. If DB2 UET cannot detect any threads that hold locks on the index space, DB2 UET cancels threads that hold locks on the table space that contains the table on which the index is created.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the table space that is associated with the specified index space to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

Tip: You can also specify the optional PART parameter to cancel threads that reference specific partitions of the index space.

JOBNAME *job_name*
Use this parameter to cancel threads based on the name of the job from which the threads originated. A job name can be up to eight characters long. Wildcards are permitted.

ORIGINAL_AUTHID *original_authorization_id*
Use this parameter to cancel threads based on the original authorization identifier of the process for which the threads were initially established. This ID can be up to eight characters long. Wildcards are permitted.

PACKAGENAME *package_name*
Use this parameter to cancel threads based on the name of the DB2 package that they are running under. A package name can be up to 8 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. Wildcards are permitted.

PART *integer*
Use this parameter to cancel threads based on the partitions in an index space or table space that the threads are referencing. Wildcards are *not* permitted. You must specify the partition numbers.

A partition number can be an integer from 1 through 4096 for DB2 Version 8 or later. Specify multiple partitions as follows:

- A series of individual partitions: *integer1*,*integer2*,*integer3*
- A range of partitions: *integer1*:*integer3* (The first value must be less than the second value.)
- Both ranges and individual partitions: *integer1*,*integer2*:*integer4*,*integer5* (The first value must be less than the second value.)

If you specify this parameter, you must also specify the INDEXSPACE or TABLESPACE parameter. To indicate partitions of a specific configuration of an index space or a table space, also specify a database name.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the table space that contains the specified partitions to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

PLAN_DEPENDENCY_CHECK | **YES** | **NO**
Use this parameter either to cancel threads for plan and package dependent objects of the table space defined in the CANCEL_THREAD syntax, or to bypass the check and only cancel and block threads that apply to the object specified. The default is YES. By specifying NO, plan and package dependency checking is skipped during thread discovery.

PLANNAME *plan_name*
Use this parameter to cancel threads based on the name of the DB2 plan that they are running under. A plan name can be up to eight characters long. Wildcards are permitted.

PROGRAM *program_name*
Use this parameter to cancel threads based on the name of a program that is associated with the threads. A program name can be up to eight characters long. Wildcards are permitted. DB2 UET cancels all threads running under a database request module (DBRM) that has a name matching the specified program name.

SYNONYM [*creator_id***.]***synonym_name*
Use this parameter to cancel threads based on the DB2 synonyms for the tables that the threads are referencing. You can specify either the synonym name only or both the synonym name and the creator ID for the synonym. When specifying both the synonym name and creator ID, use the format *creator_id*.*synonym_name*. A synonym name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. A creator name can be up to 128 characters long. Wildcards are permitted in both parts of the value. If you omit the creator ID, DB2 UET cancels threads for all synonyms that match the specified synonym name, regardless of their creator IDs.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the table space that contains the tables to which the synonym refers to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

TABLE [*creator_id*.]*table_name*
Use this parameter to cancel threads that are referencing a DB2 table. You can specify either the table name only or both the table name and the creator ID for the table. When specifying both the table name and creator ID, use the format *creator_id.table_name*. A table name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. A creator name can be up to 128 characters long. Wildcards are permitted in both parts of the value. If you omit the creator ID, DB2 UET cancels threads for all tables that match the specified table name, regardless of their creator IDs.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the table space that contains the specified table to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

TABLESPACE [*database_name*.]*table_space_name*
Use this parameter to cancel the threads that are referencing a DB2 table space or the objects in a table space. You can specify the table space name only or both the table space name and a database name in the format *database_name.table_space_name*. A table space name and a database name can each be up to eight characters long. Wildcards are permitted in each part of the value. If you omit the database name or specify only the % wildcard for the database name in this parameter, and if the DATABASE parameter does not specify a database name, DB2 UET cancels threads on all configurations of the specified table space in all databases.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the specified table space to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

Tip: You can also specify the optional PART parameter to cancel threads that reference specific partitions of the table space.

THREAD_TOKEN *nnnn*
Use this parameter to cancel a thread based on its unique, non-zero thread token value. A thread token is a 1- to 6-digit decimal number (including leading zeroes) that DB2 assigns to a thread. A token value must be unique on a DB2 subsystem or within a data sharing group. Do not specify a token value of zero for this parameter. Although threads can have token values of zero, DB2 UET does not cancel such threads, even with escalated cancelation processing.

If you specify this parameter, you cannot specify any other thread-filtering parameters. However, you can specify the control parameters.

VIEW [*creator_id*.]*view_name*
Use this parameter to cancel threads based on the DB2 views that are associated with the tables that the threads are referencing. You can specify

either the view name only or both the view name and the creator ID for the view. When specifying both the view name and creator ID, use the format *creator_id.view_name*. A view name can be up to 18 characters long in DB2 UDB for z/OS Version 7 or up to 128 characters long in later DB2 versions. A creator name can be up to 128 characters long. Wildcards are permitted in both parts of the value. If you omit the creator ID, DB2 UET cancels threads for all views that match specified view name, regardless of their creator IDs.

If you enabled thread blocking for the cancelation job, DB2 UET will change the status of the table space that contains the tables to which the view refers to either RO (read only) or UT (only the DB2 utility for which threads are being canceled has access). The status that is used depends on the ACCESS value (READ, UPDATE, or ALL) that is specified for the cancel request.

Canceling threads based on plan and package dependency

The DB2 UET cancels threads for specified objects based on locks and claims held, and plan and package dependency.

For example, the following table shows object dependencies for the plan ABPWRK00.

```
S Object Name        Owner    Type Plan Name
  *                  *        *    *
- ------------------ -------- ---- ---------
  WRK01TB1           ABPWRK01 T    ABPWRK00
  WRK01TS1           ABPWRK01 R    ABPWRK00
  WRK02TB1           ABPWRK02 T    ABPWRK00
  WRK02TS1           ABPWRK02 R    ABPWRK00
  WRK03TB1           ABPWRK03 T    ABPWRK00
  WRK03TS1           ABPWRK03 R    ABPWRK00
  WRK04TB1           ABPWRK04 T    ABPWRK00
  WRK04TS1           ABPWRK04 R    ABPWRK00
  WRK05TB1           ABPWRK05 T    ABPWRK00
  WRK05TS1           ABPWRK05 R    ABPWRK00
```

- Object Name - the name of the object that the plan is dependent upon.
- Owner - the authorization ID for the object in the column Object Name. If the object is a table space, then this field contains the name of the database.
- Type - the type of object. Valid values are: I-Index, R-Table space, S-Synonym, T-Table, and V-View.
- Plan Name - the name of the plan.

A cancel specification using the parameter ACCESS ALL, such as in the following example, causes all threads dependent upon plan ABPWRK00 to be canceled.

```
CANCEL_THREADS
(
  ACCESS ALL
  TABLESPACE ABPWRK04.WRK04TS1
)
```

For example, if 10 threads are active under plan ABPWRK00, and the previous cancel specification is used in the DB2 UET batch interface, then the following report will show referenced objects other than table space ABPWRK04.WRK04TS1 are canceled because of plan dependency, as indicated by the letter 'P' in the last position of the MATCH column.

```
** All Active Threads Objects Referenced Report for CANCEL_THREADS Request=1      PAGE=1    **
```

```
SEQNO CAN TTOKEN ELAPTIME     PLANNAME  AUTHID    MATCH  DBNAME    PSNAME    PART  LOCK_TYPE     STATE  LOCK_COUNT
----- --- ------ -----------  --------  --------  -----  --------  --------  ----- ------------  -----  ----------
    1       1331 00:00:00.25  ABPRH01   ABPSTC          **NO OBJECTS REFERENCED**
    2 ***   1352 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    3 ***   1353 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK01  WRK01TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK01  WRK01TS1      0  PAGESET       IX            1
    4 ***   1354 00:00:00.00  ABPWRK00  PDRICK    ***    ABPWRK04  WRK04TS1      0  DATA_PAGE     X             1
                                                  ***    ABPWRK04  WRK04TS1      0  PAGESET       IX            1
    5 ***   1355 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    6 ***   1356 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    7 ***   1357 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    8 ***   1358 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK03  WRK03TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK03  WRK03TS1      0  PAGESET       IX            1
    9 ***   1359 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK03  WRK03TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK03  WRK03TS1      0  PAGESET       IX            1
   10 ***   1360 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK01  WRK01TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK01  WRK01TS1      0  PAGESET       IX            1
   11 ***   1361 00:00:00.00  ABPWRK00  PDRICK    *** P  ABPWRK01  WRK01TS1      0  DATA_PAGE     X             1
                                                  *** P  ABPWRK01  WRK01TS1      0  PAGESET       IX            1
** END OF REPORT **
```

You can avoid thread cancelations based on plan and package dependency by using the parameters ACCESS UPDATE and ACCESS READ in the cancel specification. A cancel specification such as the following yields the following results.

```
CANCEL_THREADS
(
  ACCESS UPDATE
  TABLESPACE ABPWRK04.WRK04TS1
)
```

```
** All Active Threads Objects Referenced Report for CANCEL_THREADS Request=1      PAGE=1    **
```

```
SEQNO CAN TTOKEN ELAPTIME     PLANNAME  AUTHID    MATCH  DBNAME    PSNAME    PART  LOCK_TYPE     STATE  LOCK_COUNT
----- --- ------ -----------  --------  --------  -----  --------  --------  ----- ------------  -----  ----------
    1       1375 00:00:00.39  ABPRH01   ABPSTC          **NO OBJECTS REFERENCED**
    2       1386 00:00:00.00  ABPWRK00  PDRICK          ABPWRK03  WRK03TS1      0  DATA_PAGE     X             2
                                                        ABPWRK03  WRK03TS1      0  PAGESET       IX            1
    3 ***   1387 00:00:00.00  ABPWRK00  PDRICK    ***   ABPWRK04  WRK04TS1      0  DATA_PAGE     X             1
                                                  ***   ABPWRK04  WRK04TS1      0  PAGESET       IX            1
    4       1388 00:00:00.00  ABPWRK00  PDRICK          ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                        ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    5       1389 00:00:00.00  ABPWRK00  PDRICK          ABPWRK03  WRK03TS1      0  DATA_PAGE     X             1
                                                        ABPWRK03  WRK03TS1      0  PAGESET       IX            1
    6 ***   1390 00:00:00.00  ABPWRK00  PDRICK    ***   ABPWRK04  WRK04TS1      0  DATA_PAGE     X             1
                                                  ***   ABPWRK04  WRK04TS1      0  PAGESET       IX            1
    7       1391 00:00:00.00  ABPWRK00  PDRICK          ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                        ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    8       1392 00:00:00.00  ABPWRK00  PDRICK          ABPWRK02  WRK02TS1      0  DATA_PAGE     X             1
                                                        ABPWRK02  WRK02TS1      0  PAGESET       IX            1
    9       1393 00:00:00.00  ABPWRK00  PDRICK          ABPWRK01  WRK01TS1      0  DATA_PAGE     X             1
                                                        ABPWRK01  WRK01TS1      0  PAGESET       IX            1
   10       1394 00:00:00.00  ABPWRK00  PDRICK          ABPWRK03  WRK03TS1      0  DATA_PAGE     X             1
                                                        ABPWRK03  WRK03TS1      0  PAGESET       IX            1
   11       1395 00:00:00.00  ABPWRK00  PDRICK          ABPWRK03  WRK03TS1      0  DATA_PAGE     X             2
                                                        ABPWRK03  WRK03TS1      0  PAGESET       IX            1
** END OF REPORT **
```

Only threads 1387 and 1390 that were updating the specific objects are canceled, as indicated in the column titled MATCH. Plan and package dependency checking is avoided by specifying ACCESS UPDATE and ACCESS READ. Only threads on objects that directly match the object specification are canceled.

Adding comments to a job step

You can add comments among the global parameters and cancel requests in a batch job step to clarify the job syntax for yourself and other users. A comment is descriptive only. It does not affect how the job runs.

About this task

To specify a comment, type two consecutive hyphens in columns 1 and 2 followed by the comment text. Only the comment itself can occur on the line.

Example

The following example contains two comments:

```
//ABPPARMS DD   DATA
--Some comments.
  DB2SSID(DBP2)
  EXEC_TYPE(EXECUTE)
  REPORT_TYPE(DETAIL)
  CANCEL_THREADS(
    DATABASE AB100DB
       )
--Some more comments.
  CANCEL_THREADS(
    PLANNAME DSNESPCS
       )
/*
//
```

Examples of thread-cancel and thread-blocker job steps

Review these example jobs to learn how you might code your thread-cancel jobs with and without thread blocking. The examples demonstrate the use of various input parameters.

Example 1

The following example contains two cancel requests. The first cancel request uses the DATABASE parameter with the % wildcard (representing zero or more characters) to cancel threads that are referencing objects in any DB2 database that has the name "PAY" or a name beginning with "PAY." The second cancel request uses the TABLESPACE parameter with a value that specifies both a database name and a table space name that includes the _ wildcard (representing a single character); this cancel request will cancel threads on table spaces in the database MYDB0001 that have names beginning with "MYTS000" and ending with any character. Because the REPORT_TYPE parameter is set to DETAIL, the job output will include all of the reports that DB2 UET can generate for a thread-cancellation job.

```
//JOBNAME JOB ,'MYNAME',CLASS=A,MSGCLASS=X
//*
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000
//STEPLIB  DD   DISP=SHR,DSN=ABP.INSTALL.SABPLOAD
//*
//ABPPARMS DD   *
  ABPID (ABP01)
  DB2SSID (DSN1)
  EXEC_TYPE (EXECUTE)
  REPORT_TYPE (DETAIL)
  CANCEL THREADS (
    DATABASE PAY%
  )
  CANCEL THREADS (
    TABLESPACE MYDP0001.MYTS000_
  )
/*
```

Example 2

The following example contains a single cancel request. Because no ABPID parameter is specified, a DB2 UET configuration on DSN1 will be automatically selected to perform cancelation processing. DB2 UET will perform a trial run of canceling threads because the EXEC_TYPE parameter is set to SIMULATE. The trial run will simulate the cancelation of only the distributed database access threads that originate from the specified DDF requestor IP address. Also, even though the REPORT_TYPE parameter is not specified, DB2 UET will generate the two reports for "canceled" threads because the default report type of SUMMARY is used.

```
//JOBNAME JOB ,'MYNAME',CLASS=A,MSGCLASS=X
//*
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000
//STEPLIB  DD   DISP=SHR,DSN=ABP.INSTALL.SABPLOAD
//*
//ABPPARMS DD   *
  DB2SSID (DSN1)
  EXEC_TYPE (SIMULATE)
  CANCEL THREADS (
    DDF_DBAT_REQUESTOR_IPADDRESS 100.10.10.1
  )
/*
```

Example 3

The following example contains a single cancel request. Because the EXEC_TYPE parameter is not specified, the default execution type of EXECUTE is used. DB2 UET will perform an escalated cancelation of any threads that are running under a plan that originate from a job named CICS0001.

```
//JOBNAME JOB ,'MYNAME',CLASS=A,MSGCLASS=X
//*
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000
//STEPLIB  DD   DISP=SHR,DSN=ABP.INSTALL.SABPLOAD
//*
//ABPPARMS DD   *
  DB2SSID (DSN1)
  ABPID (ABP03)
  CANCEL THREADS (
    JOBNAME CICS0001
    ESCALATE YES
  )
/*
```

Example 4

The following example cancels existing threads and blocks new threads on two DB2 table spaces to ensure that these table spaces are available to the utilities that need them. To implement thread blocking, you must create two job steps: one to block and cancel threads and one to allow new threads to form again. In between these DB2 UET job steps, you run the job steps for the utilities.

In each of the DB2 UET job steps, you must specify a PARM in the EXEC statement, the ABPID parameter, and the EXEC_TYPE (EXECUTE) parameter.

In the BLOCK_THREADS job step only, you must also include a CANCEL_THREADS request with the parameters for the table spaces. Optionally, you can also specify the THREAD_QUIESCE_TIME and ON_FAILURE parameters. The THREAD_QUIESCE_TIME parameter indicates that DB2 UET should wait 30

seconds between the initiation of thread blocking and the cancelation of threads to enable the applications that are using the existing threads to complete their work. The ON_FAILURE parameter controls what happens when a cancelation error occurs; in this case, the job will terminate and the statuses of the table spaces will be reset to their original values.

The ALLOW_THREADS job step does not require any cancel request or cancel parameters. However, it must have the same *blocker_id* as the ALLOW_THREADS job step.

The following job step blocks and cancels threads on two table spaces:

```
//JOBNAME JOB ,'MYNAME',CLASS=A,MSGCLASS=X
//*
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000,
//         PARM='MYBLOCKER,BLOCK_THREADS'
//STEPLIB  DD   DISP=SHR,DSN=ABP.INSTALL.SABPLOAD
//*
//ABPPARMS DD   *
  DB2SSID (DSN1)
  ABPID (ABP03)
  EXEC_TYPE (EXECUTE)
  ON_FAILURE (TERMINATE RESET_OBJECT_STATUS YES)
  THREAD_QUIESCE_TIME (30)
  CANCEL THREADS (
    DATABASE PAY123
    TABLESPACE MYTS0001
  )
  CANCEL THREADS (
    DATABASE PAY456
    TABLESPACE MYTS0009
  )
/*
```

The following job step allows new threads to form once again on the table spaces:

```
//JOBNAME JOB ,'MYNAME',CLASS=A,MSGCLASS=X
//*
//EXECABP  EXEC PGM=ABPBMAIN,REGION=0M,DYNAMNBR=1000,
//         PARM='MYBLOCKER,ALLOW_THREADS'
//STEPLIB  DD   DISP=SHR,DSN=ABP.INSTALL.SABPLOAD
//*
//ABPPARMS DD   *
  DB2SSID (DSN1)
  ABPID (ABP03)
  EXEC_TYPE (EXECUTE)
/*
```

Running a thread-cancelation job in a simulation mode

You can first run a thread-cancelation job in a simulation mode to check the job syntax and potential results. When you are satisfied with the job setup, you can run the job in execution mode to actually cancel threads.

About this task

EXECUTE mode is the default if you do not specify an EXEC_TYPE value.

Procedure

1. To check the job syntax only, set the EXEC_TYPE parameter to CHECKPARMS, and then run the job. CHECKPARMS validates the input parameters. No simulated or actual thread-cancelation processing is performed. The job output, which is written to SPRT0000, will indicate any syntax errors.

2. To perform a trial run of the job so that you can preview the results, set the EXEC_TYPE parameter to SIMULATE, and then run the job. SIMULATE performs a trial run of the job so that you can check that your thread-filtering criteria selects the correct set of threads to cancel and troubleshoot any potential problems.DB2 UET displays the processing flow, messages, and reports for all cancel requests in the job but does not actually cancel any threads. You can review the output to determine whether the job will complete successfully and if the thread-filtering criteria that you specified will select the correct set of threads to cancel. No thread blocker activity is reported in the simulated output.
3. When you are satisfied that the JCL is correct, set the EXEC_TYPE parameter to EXECUTE, and then either submit the job as a standalone job or include it as a job step within a batch job that runs during the batch window. EXECUTE actually runs the job. Threads that match the specified thread-filtering parameters are canceled.

Determining whether thread blocking and canceling occurred

DB2 UET generates several types of output for a thread-cancel or thread-blocker job step. Check this output to determine if your job step completed successfully.

You can view any of the following types of output by using SDSF or an equivalent tool:

- Messages that are issued from the batch interface for the thread-cancelation job step and each cancel request in the job step. These messages have message numbers that begin with "ABPB." For descriptions of these messages, see “DB2 Utilities Enhancement Tool messages” on page 375
- Summaries of input parameter processing and cancel request processing (the SPRT0000 and SPRT*nnnn* data sets in the job output).
- The reports on the threads canceled. If you set the REPORT_TYPE global parameter to DETAIL, the reports on all active threads are also generated.
- Return codes for the job step as a whole and for individual cancel requests.

  You can use the return codes that the DB2 UET batch interface returns for a thread-cancelation job step to control whether subsequent applications and programs in the batch job run. For example, you might want to prevent the execution of subsequent programs when return code 28 is issued. This return code indicates that a thread blocking or cancelation operation failed. If you allowed the subsequent programs to run, they might not be able to access the DB2 objects that they need because threads on those objects were not canceled.For descriptions of these codes, see “DB2 Utilities Enhancement Tool codes” on page 436.

If you need to contact IBM Software Support, see “Diagnostic information for Support” on page 439.

Related reference:

“DB2 Utilities Enhancement Tool messages” on page 375
Look up DB2 UET messages to obtain information about them, including message explanations and suggested responses.

“DB2 Utilities Enhancement Tool codes” on page 436
Review the descriptions of the return codes and abend codes that can be issued from the DB2 Utilities Enhancement Tool started task, DSNUTILB intercept, or batch interface to determine how DB2 Utilities Enhancement Tool processing ended. For descriptions of any DB2 codes that might be returned for DB2 Utilities Enhancement Tool processing, see the *DB2 for z/OS Codes* manual.

Summary of input parameter processing

DB2 UET generates one SPRT0000 data set for a thread-cancelation job. This data set lists the global parameters and the CANCEL_THREADS parameters that were defined for the job and indicates whether they were processed successfully.

The following figure shows a sample of the summary output:

```
******************************** TOP OF DATA ********************************************************************
BPB6100I DB2 Utilities Enhancement Tool 02.20 execution has started
ABPB6040I Processing of input parameters has started
ABPP9910I date time ABPID(ABP1)
ABPP9910I date time DB2SSID(DBP1)
ABPP9910I date time ESCAPE(\)
ABPP9910I date time EXEC_TYPE(EXECUTE)
ABPP9910I date time REPORT_TYPE(DETAIL)
ABPP9910I date time CANCEL_THREADS(
ABPP9910I date time AUTHID PDUSR1
ABPP9910I date time CONNTYPE TSO
ABPP9910I date time )
ABPB6042I Processing of input parameters ended successfully
ABPB6050I Initialization is complete. DB2SSID=DBP1  VERSION=10.1.0  NETID=ABCDNET1  LUNAME=D8CDB2  LOCATION=RS22D8A
ABPB6700I DB2 Utilities Enhancement Tool execution ended. Highest RC=00000000*
******************************** BOTTOM OF DATA *****************************************************************
```

Figure 26. Sample output for input parameter processing (SPRT0000)

If input parameter processing did not complete successfully, you should check the JCL for syntax errors.

Summary of cancel request processing

DB2 UET generates one SPRT*nnnn* data set for each CANCEL_THREADS request in a job. This data set summarizes the processing that occurred for each thread that was selected for cancelation based on the thread-filtering parameters that were specified in the cancel request.

You can use this summary to obtain the following information about the cancel request:

- All of the thread-filtering parameters that were specified
- If the execution mode was SIMULATE
- The total number of active threads
- The number of threads that matched the thread-filtering parameters
- Whether the cancel command (the DB2 -CANCEL THREAD or escalated cancel command) was issued for each thread of interest
- The termination-checking activity that occurred
- Whether the thread actually terminated
- The highest return code for the cancel request

The following figure shows a sample of this output:

```
******************************** TOP OF DATA ***************************************************************************
ABPB6120I Processing CANCEL_THREADS request=1   , Started at 14:10:00.8; Parameters are as follows:
ABPB6999I     THREAD_TOKEN 0305
ABPB6410I Thread list build was successful: Active thread count=3
ABPB6411I Thread filtering applied: Threads of cancel interest count=1
ABPB6450I Thread cancel issued:  PLAN=DSNESPRR  CONN=TSO       CORR=PDABCD        AUTH=PDABCD    OAUTH=PDABCD    TOKN=305
ABPB6451I                        JOBN=PDABCD    ASID=00BF      PROGNAME=DSNESM68  COLLID=DSNESPRR
ABPS0220I TCB: 007CCE88 Session: 18925F10 - CANCEL THREAD requested for thread token 305
ABPS0211I  DSNV426I  !D7B DSNVCT THREAD '000305' HAS BEEN CANCELED
ABPB6470I Thread status checking for canceled threads started. Retry count=5     Retry interval=2
ABPB6462I Thread *still active*: PLAN=DSNESPRR  CONN=TSO       CORR=PDABCD        AUTH=PDABCD    OAUTH=PDABCD    TOKN=305
ABPB6473I Thread status checking is being retried. Threads still active =1     Retries remaining=4
ABPB6461I Thread has terminated: PLAN=DSNESPRR  CONN=TSO       CORR=PDABCD        AUTH=PDABCD    OAUTH=PDABCD    TOKN=305
ABPB6471I Thread status checking ended; all canceled threads have terminated
ABPB6121I Processing of CANCEL_THREADS request=1    has ended at 14:10:03.2: Threads canceled    , RC=00000000
ABPB6700I DB2 Utilities Enhancement Tool cancel execution ended. Highest RC=00000000

******************************** BOTTOM OF DATA ************************************************************************
```

Figure 27. Sample output for CANCEL_THREADS request processing (SPRTnnnn)

This sample indicates that DB2 UET identified one thread to cancel based on the single thread-filtering parameter THREAD_TOKEN. The return code indicates that the cancel request ended successfully.

Reports

DB2 UET generates reports for each cancel request in a thread-cancelation job. The reports contain information about the threads that were canceled and the threads that were active when the job ran. Print and keep these reports for your records.

To control whether all reports are generated or only those on threads canceled, specify a value for the REPORT_TYPE global parameter in the thread-cancelation job as follows:

- SUMMARY generates the Threads Canceled Report and the Threads Canceled Unit of Recovery Report, provided that some threads matched the selection criteria for cancel processing.
- DETAIL generates all of the following reports, provided that some threads were selected for cancelation:
  - Threads Canceled Report
  - Threads Canceled Unit of Recovery Report
  - All Active Threads Report
  - All Active Threads Unit of Recovery Report
  - All Active Threads Objects Referenced Report

  If no threads were selected for cancelation, only the three reports on all active threads are generated.

Note that DB2 UET also generates these reports for any thread-cancelation processing that you perform by using the DSNUTILB intercept.

If you do not specify the REPORT_TYPE parameter in a batch job or in the *abpid*BGLB member for the DSNUTILB intercept (where *abpid* is the started task instance ID), DB2 UET generates only the SUMMARY reports by default.

The all active threads reports present data for all active threads in your environment at the time a cancel request is processed. The threads that these reports cover might vary for different cancel requests. Because the cancel requests are processed sequentially, any active threads that complete or are canceled while a cancel request is being processed are not included in the all active threads reports for the next cancel request.

Tip: Most of the thread information that the reports display is obtained from DB2. Refer to the IBM DB2 documentation for detailed information about these fields.

Threads Canceled Report:

The Threads Canceled Report provides information about the threads that were selected for cancelation processing in a cancel request. Each report column is described.

The following figure shows a sample report:

```
******************************** TOP OF DATA ********************************************************************************

** Threads Canceled Report for CANCEL_THREADS Request=1      PAGE=1    **
_____________________________________________________________________

SEQNO TTOKEN ELAPTIME/   PLANNAME/    CONNID/      AUTHID/  JOBNAME/ PROGRAM  INDB2 TIME/     COMMIT/  SQL CALL/
             STATUS/     CORRID       CONNTYPE     ORIGAUTH ASIDX             INDB2 CPU       ABORT    GETPAGE
             COLLID
----- ------ ----------- ------------ ------------ -------- -------- -------- --------------- -------- ----------
    1   1781 00:00:00.11 PGMAPLAN     TSO          PDUSR1   PDUSR1   DSNESM68 00:00:00.119389        1         15
             T           PDUSR1       TSO          PDUSR1   03F1              00:00:00.002683        0          4
             DSNESPRR
    2   1783 00:00:00.00 PGMAPLAN     TSO          PDUSR1   PDUSR1   DSNESM68 00:00:00.002280        1         15
             T           PDUSR1       TSO          PDUSR1   03F1              00:00:00.001707        0          4
             DSNESPRR
** END OF REPORT **
******************************** BOTTOM OF DATA *****************************************************************************
```

Figure 28. Sample Threads Canceled Report (REPTnnnn)

Related reference:

"Report columns" on page 239
This section provides an alphabetical list and descriptions of the report columns, and indicates the reports that contain each column.

Threads Canceled Unit of Recovery Report:

The Threads Canceled Unit of Recovery Report provides information about the current unit of recovery (UR), if any, for each thread that was selected for cancelation processing in a cancel request. Each report column is described.

The report includes information on when the UR started and its current status. If no UR existed or if no UR information was available for a thread, the report displays the text **NO UR** under the **UR_STRT_DT** column, and no UR data appears on the right-hand side of that column.

The following figure shows a sample report:

```
******************************** TOP OF DATA ********************************************************************************
** Threads Canceled Unit of Recovery Report for CANCEL_THREADS Request=1      PAGE=1    **
_____________________________________________________________________

SEQNO TTOKEN PLANNAME AUTHID   UR_STRT_DT UR_STRT_TIME    UR_STATE UR_ELAP_TIME    UR_LOG_SRBA/         LOG_PAGES LOG_RECS
                                                                                  UR_LOG_ERBA
----- ------ -------- -------- ========== ------------- -------- --------------- -------------------- --------- ---------
    1    387 ABPDKCYP PDKILI   12/10/2015 13:18:12.187296          00:00:11.044358 000000000002C58011DF     50632   2000152
                                                                                  000000000002D1DC8F28
** END OF REPORT **
******************************** BOTTOM OF DATA *****************************************************************************
```

Figure 29. Sample Threads Canceled Unit of Recovery Report (REPTnnnn)

Related reference:

"Report columns" on page 239
This section provides an alphabetical list and descriptions of the report columns,

and indicates the reports that contain each column.

All Active Threads Report:

The All Active Threads Report provides information about each active thread on the DB2 subsystem, including whether the thread was selected for cancelation processing and its current status in DB2. Each report column is described.

The following figure shows a sample report:

```
******************************** TOP OF DATA*********************************************************************************
 ** All Active Threads Report for CANCEL_THREADS Request=2      PAGE=1    **
 ____________________________________________________________________

 SEQNO CAN TTOKEN ELAPTIME/   PLANNAME/    CONNID/      AUTHID/  JOBNAME/ PROGRAM  INDB2 TIME/     COMMIT/  SQL CALL/
                  STATUS/     CORRID       CONNTYPE     ORIGAUTH ASIDX             INDB2 CPU       ABORT    GETPAGE
                  COLLID
 ----- --- ------ ----------- ------------ ------------ -------- -------- -------- --------------- -------- ----------
     1       1779 00:00:00.20 ABPXX01      DB2CALL      ABPSTC   ABPXX01           00:00:00.202382        1         73
                  T           ABPXX01      CAF          ABPSTC   00C1              00:00:00.084015        0          0

     2       1762 00:00:00.39 ABPYY01      DB2CALL      ABPSTC   ABPYY01           00:00:00.393813        1        211
                  T           ABPYY01      CAF          ABPSTC   03EB              00:00:00.190803        0          0

     3 ***   1760 00:00:00.10 ABPZZ01      DB2CALL      ABPSTC   ABPZZ01           00:00:00.109812        1         29
                  T           ABPZZ01      CAF          ABPSTC   03EA              00:00:00.031711        0          0

 ** END OF REPORT **
******************************** BOTTOM OF DATA *****************************************************************************
```

Figure 30. Sample All Active Threads Report (REPTnnnn)

Related reference:

"Report columns" on page 239
This section provides an alphabetical list and descriptions of the report columns, and indicates the reports that contain each column.

All Active Threads Unit of Recovery Report:

The All Active Threads Unit of Recovery Report provides information about the current unit of recovery (UR), if any, for each active thread on the DB2 subsystem. Each report column is described.

The report includes information on when the UR started and its current status. If no UR exists or if no UR information is available for a thread, the report displays the text **NO UR** under the **UR_STRT_DT** column, and no UR data appears to the right-hand side of that column.

The following figure shows a sample report:

```
******************************** TOP OF DATA ********************************************************************************

** All Active Threads Unit of Recovery Report for CANCEL_THREADS Request=1      PAGE=1    **
_____________________________________________________________________________________

SEQNO CAN TTOKEN PLANNAME AUTHID   UR_STRT_DT UR_STRT_TIME    UR_STATE UR_ELAP_TIME    UR_LOG_SRBA/         LOG_PAGES LOG_RECS
                                                                                      UR_LOG_ERBA
----- --- ------ -------- -------- ========== --------------- -------- --------------- -------------------- --------- ---------
    1        987 DK27PLN  ABPSTC   12/06/2015 08:12:14.254096          00:00:16.223643 0000000000003080A908         1        42
                                                                                      0000000000003080AD05

    2        988 DK27PLN  ABPSTC   12/06/2015 08:12:14.259392          00:00:16.220520 0000000000003080B72E         1        82
                                                                                      0000000000003080B9AF

** END OF REPORT **
******************************** BOTTOM OF DATA *****************************************************************************
```

Figure 31. Sample All Active Threads Unit of Recovery Report (REPTnnnn)

Related reference:

"Report columns"
This section provides an alphabetical list and descriptions of the report columns, and indicates the reports that contain each column.

All Active Threads Objects Referenced Report:

The All Active Threads Objects Referenced Report provides information about the objects that were referenced by each active thread on the DB2 subsystem. Each report column is described.

The report includes the DB2 database name, the page set name, the partition identifier, and the type of lock held on the object. If a thread was active but not referencing any objects, the report displays the text **NO OBJECTS REFERENCED** under the **DBNAME**, **PSNAME**, and **PART** columns, and no data appears to the right-hand side of those columns.

The following figure shows a sample report:

```
******************************** TOP OF DATA ***************************************************************************

** All Active Threads Objects Referenced Report for CANCEL_THREADS Request=1      PAGE=1    **
_____________________________________________________________________________________________

SEQNO CAN TTOKEN ELAPTIME     PLANNAME  AUTHID    MATCH  DBNAME    PSNAME    PART  LOCK_TYPE     STATE  LOCK_COUNT
----- --- ------ -----------  --------  --------  -----  --------  --------  ----- ------------  -----  ----------
    1        457 00:00:00.00  PGMAPLAN  PDUSR1           **NO OBJECTS REFERENCED**
    2 ***    456 00:00:00.00  PGMBPLAN  PDUSR2           ABP100DB  EMPLOYEE      0  PAGESET       IS            1
                                                         DSNDB07   DSN4K01       0  PAGESET       S             1
    3        431 00:00:00.09  PGMAPLAN  PDUSR1           **NO OBJECTS REFERENCED**
    4        424 00:00:00.09  PGMCPLAN  PDUSRX           DSNDB06   SYSUSER       0  DATA_PAGE     S             1
                                                         DSNDB06   SYSUSER       0  PAGESET       IS            1
                                                         ABPTST02  NUMPARTS      1  DATA_PAGE     X             1
                                                         ABPTST02  NUMPARTS      4  PART_TS       IX            1
    5        422 00:00:00.40  PGMAPLAN  PDUSR1           **NO OBJECTS REFERENCED**
    6        455 00:00:02.75  PGMBPLAN  PDUSR2           DSNDB06   SYSDBASE      0  PAGESET       IS            1
                                                         DSNDB06   SYSDBAUT      0  PAGESET       IS            1
                                                         DSNDB06   SYSUSER       0  PAGESET       IS            1
                                                         AB100DB   AB100TS1      0  PAGESET       IS            1
** END OF REPORT **
******************************** BOTTOM OF DATA ************************************************************************
```

Figure 32. Sample All Active Threads Objects Referenced Report (REPTnnnn)

Related reference:

"Report columns"
This section provides an alphabetical list and descriptions of the report columns, and indicates the reports that contain each column.

Report columns:

This section provides an alphabetical list and descriptions of the report columns, and indicates the reports that contain each column.

Column	Reports that contain this column
ABORT The number of DB2 abort operations that the thread performed.	Threads Canceled All Active Threads
ASIDX The address space identifier (in hexadecimal format) for the address space from which the thread originated.	Threads Canceled All Active Threads

Column	Reports that contain this column
AUTHID The current authorization ID of the process that was associated with the thread. (*Threads Canceled, All Active Threads only*): If you used the SQL statement SET CURRENT SQLID to change the current authorization ID, this field displays the new authorization ID and the ORIGAUTH field displays the original authorization ID.	Threads Canceled Threads Canceled Unit of Recovery All Active Threads All Active Threads Unit of Recovery All Active Threads Objects Referenced
CAN Whether DB2 UET selected the thread for cancelation processing. If the thread was selected for cancelation, the field displays three asterisks (***).	All Active Threads All Active Threads Unit of Recovery All Active Threads Objects Referenced
COLLID The collection identifier for the group of packages that includes the package under which the thread was running.	Threads Canceled All Active Threads
COMMIT The number of DB2 commits that the thread performed.	Threads Canceled All Active Threads
CONNID The connection identifier that is supplied by the attachment facility and that is associated with the specific address space from which the thread originated.	Threads Canceled All Active Threads
CONNTYPE The type of DB2 connection, which can be one of the following values: • CAF: The thread originates from a Call Attach Facility (CAF) user. • CICS: The thread originates from a CICS region. • DBAT_DRD: The thread is a distributed database access thread that uses distributed relational database access protocols (a DRDA connection). • DBAT_PP: The thread is a distributed database access thread that uses distributed database private protocols (a DDF connection). • IMSBMP: The thread originates from an IMS Batch Message Processing (BMP) program. • IMSCTL: The thread originates from an IMS control region. • IMSDLIBT: The thread originates from an IMS Data Language/I (DL/I) batch region. • IMSMPP: The thread originates from an IMS message processing program. • IMSTBMP: The thread originates from an IMS transaction BMP. • RRSAF: The thread originates from the Recoverable Resource Manager Service attachment facility. • TSO: The thread originates from a TSO/E interactive or batch user. • UTILITY: The thread originates from a DB2 utility.	Threads Canceled All Active Threads

Column	Reports that contain this column
CORRID The correlation identifier for the thread. For example, this ID can be a TSO user ID or a job name.	Threads Canceled All Active Threads
DBNAME The name of the database that contained the referenced DB2 object.	All Active Threads Objects Referenced
ELAPTIME The amount of time that elapsed between thread creation or sign-on to DB2 and • (*Threads Canceled only*) thread termination or report generation • (*All Active Threads, All Active Threads Objects Referenced only*) report generation This time is in the format HH:MM:SS:xx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xx* is hundredths of a second.	Threads Canceled All Active Threads All Active Threads Objects Referenced
GETPAGE The number of DB2 getpages operations that were performed on behalf of the thread.	Threads Canceled All Active Threads
INDB2 CPU The amount of CPU time that elapsed while the thread was running in DB2. This interval is in the format HH:MM:SS:xxxxxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxxxxx* is microseconds.	Threads Canceled All Active Threads
INDB2 TIME The amount of wall-clock time that elapsed while the thread was running in DB2. This interval is in the format HH:MM:SS:xxxxxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxxxxx* is microseconds.	Threads Canceled All Active Threads
JOBNAME The name of the job from which the thread originated.	Threads Canceled All Active Threads
LOCK_COUNT The number of DB2 locks held on the referenced object.	All Active Threads Objects Referenced
LOCK_TYPE The type of DB2 lock on the referenced object, which can be one of the following values: • PAGESET: A page set lock • PART_SPL: Selective partition or partition page set lock • PART_NSP: Non-selective partition page set lock • DATA_PAGE: Data page lock • ROW: Row level lock • INDEX_PAGE: Index page lock • MASS_DELETE: Mass delete lock • DRAIN_CSREAD: Cursor stability read drain lock • DRAIN_RRREAD: Repeatable read drain lock • DRAIN_WRITE: Write drain lock For descriptions of these options, refer to your DB2 documentation.	All Active Threads Objects Referenced

Column	Reports that contain this column
LOG_PAGES The number of physical log pages that contained information for the current UR. If no UR existed or no UR information was available for a thread, this field displays a zero.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery
LOG_RECS The number of log records that were created for the active UR. A single log page is composed of multiple log records. If no UR existed or no UR information was available for a thread, this field displays a zero.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery
MATCH Whether a DB2 object matched any thread-filtering parameters for DB2 objects that were specified for the cancel request. If a match is found, three asterisks (***) are displayed in this column.	All Active Threads Objects Referenced
ORIGAUTH The original authorization ID of the process for which the thread was initially established.	Threads Canceled All Active Threads
PART The identifier for the page set partition that was referenced or contained the referenced object.	All Active Threads Objects Referenced
PLANNAME The name of the DB2 plan under which the thread is or was running.	Threads Canceled Threads Canceled Unit of Recovery All Active Threads All Active Threads Unit of Recovery All Active Threads Objects Referenced
PROGRAM The name of the program that was associated with the thread.	Threads Canceled All Active Threads
PSNAME The name of the page set that is referenced. This name is a table space name or an index space name.	All Active Threads Objects Referenced
SEQNO The sequence number of a report line.	Threads Canceled Threads Canceled Unit of Recovery All Active Threads All Active Threads Unit of Recovery All Active Threads Objects Referenced
SQL CALL The number of SQL calls that the thread performed.	Threads Canceled All Active Threads

Column	Reports that contain this column
STATE The state of the DB2 lock on the referenced object, which can be one of the following values: • S: Share. The lock owner and any concurrent processes can read but not change data in the table space, partition, or table. • U: Update. The lock owner can read but not change the locked data. To change the data, the lock owner can promote a U lock to an X lock. Concurrent processes can acquire S locks but not U locks. U locks are intended to reduce the chance of deadlocks. • X: Exclusive. The lock owner can read or change data in the table space, partition, or table. Concurrent processes can access the data under limited conditions. • SIX: Share with intent exclusive. The lock owner can read or change data in the table space, partition, or table. Concurrent processes can read data in the table space, partition, or table, but cannot change the data. • IS: Intent share. The lock owner can read data in the table space, partition, or table, but cannot change the data. Concurrent processes can read and change the data. • IX: Intent exclusive. The lock owner and concurrent processes can read and change data in the table space, partition, or table. • NSU: Non-share update. For more information about these lock types, see the *DB2 for z/OS Administration Guide*.	All Active Threads Objects Referenced
STATUS The DB2 connection status code. Many codes are available and are described in the message DSNV404I in the *DB2 for z/OS Messages and Codes* manual.	Threads Canceled All Active Threads
TTOKEN The thread token value. A thread token value is a unique numeric identifier that DB2 assigns to each active thread on a DB2 subsystem. A token value must be unique on a subsystem or within a data sharing group. DB2 UET does not report information for threads that have a thread token value of zero.	Threads Canceled Threads Canceled Unit of Recovery All Active Threads All Active Threads Unit of Recovery All Active Threads Objects Referenced
UR_ELAP_TIME The amount of time that elapsed between the start of the UR and the retrieval of information for this report. This time is in the format HH:MM:SS:xxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxx* is milliseconds.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery
UR_LOG_ERBA The log end RBA for the active UR. This value is the RBA of the last log record written by the active UR.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery
UR_LOG_SRBA The log start relative byte address (RBA) for the active UR.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery

Column	Reports that contain this column
UR_STATE The status of the UR, which can be one of the following values: • INFLIGHT: The UR is an active, inflight unit of work. • IN PHAS1: The UR is currently in phase 1 of commit processing. • IN PHAS2: The UR is currently in phase 2 of commit processing. • COMMITED: Commit processing has ended. • IN ABORT: The UR is being aborted. Rollback processing has begun but has not yet completed. • ABORTED: The UR has been aborted. Rollback processing is complete. • INDOUBT: The UR is in the in doubt state.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery
UR_STRT_DT The start date (or creation date) of the UR. A UR exists only if the thread modified DB2 objects or resources. The date is in the format MM/DD/YYYY, where *MM* is the month, *DD* is the day, and *YYYY* is the year.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery
UR_STRT_TIME The start time (or creation time) of the UR. This time is in the format HH:MM:SS:xxx, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *xxx* is milliseconds.	Threads Canceled Unit of Recovery All Active Threads Unit of Recovery

Related reference:

"Threads Canceled Report" on page 237
The Threads Canceled Report provides information about the threads that were selected for cancelation processing in a cancel request. Each report column is described.

"Threads Canceled Unit of Recovery Report" on page 237
The Threads Canceled Unit of Recovery Report provides information about the current unit of recovery (UR), if any, for each thread that was selected for cancelation processing in a cancel request. Each report column is described.

"All Active Threads Report" on page 238
The All Active Threads Report provides information about each active thread on the DB2 subsystem, including whether the thread was selected for cancelation processing and its current status in DB2. Each report column is described.

"All Active Threads Unit of Recovery Report" on page 238
The All Active Threads Unit of Recovery Report provides information about the current unit of recovery (UR), if any, for each active thread on the DB2 subsystem. Each report column is described.

"All Active Threads Objects Referenced Report" on page 239
The All Active Threads Objects Referenced Report provides information about the objects that were referenced by each active thread on the DB2 subsystem. Each report column is described.

Chapter 6. Monitoring DB2 utility statements for text strings, events, and messages

Element <PRACTICE> enables the DB2 UET DSNUTILB intercept component to search the provided DB2 utility statements and perform actions that you specify.

By specifying subordinate elements, you can use <PRACTICE> to

- Add a text string to the utility statement if the specified string is not found in the statement syntax. The text string is appended to the end of the utility statement.
- Remove a text string from the utility statement if the specified string is found in the utility statement.
- Substitute a text string to replace the text string found in the utility statement.
- Journal the occurrence of a monitored event to a DB2 table for subsequent review or analysis.
- Fail the utility job step with a specified return code if a text string is found in the utility statement.
- Change the return code of a job based on the presence of a DB2 message. (The MESSAGE element was added to the Policy PRACTICE in DB2 UET V2.2.)

Restriction: Utility Monitor actions apply to native DB2 utility statements only. You cannot use the Utility Monitor to manipulate DB2 UET extended syntax options (such as PRESORT).

Topics:

How the Utility Monitor works

The DB2 UET has a policy that governs the behavior of the DB2 UET DSNUTILB intercept component. The rules that are defined within the policy dictate which utilities are monitored as well as what syntax to enforce.

When a DSNUTILB utility job is run, DB2 UET can gain control and evaluate the utility statement syntax along with the Utility Monitor rules within the intercept policy. Each rule within the policy relates some utility statement text with a coded option.

When the rule is evaluated as true, the option directs DB2 UET to perform one or a combination of the actions. For more information about the actions, see "<SYNTAX>" on page 151 and "<MESSAGE>" on page 138.

Using policy rules to invoke the Utility Monitor

When the DB2 UET is installed and customized, a policy with default rules is created in the SABPSAMP library. The policy is used by the started task when the DSNUTILB intercept is invoked.

The policy rules dictate how DB2 UET intercepts a DSNUTILB utility and the actions it performs. The default policy as shipped does not cause threads to be

blocked and canceled, nor does it invoke the Utility Monitor. It must be customized before any action is taken by the DB2 UET started task.

By default, the name of the policy is created as *abpid*PLCY, where *abpid* is the 4-character identification for your installation of DB2 UET. For example, an ABPID could be named ABP1 and the policy name would be ABP1PLCY.

If any of the rules within the policy evaluate as true, the action described by the policy rule will invoke the Utility Monitor.

For more information, see "Example DSNUTILB intercept policies" on page 169.

Example policy to invoke the Utility Monitor

The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

After the rules are defined, the element <DB2SYSTEM> must be specified to use the ACTION MONITOR_UTILITY; otherwise the Utility Monitor will not be invoked. For a complete DB2 UET policy that contains the practice rules that are defined in the following example, see "Example policy that details RULE and RULESET sections" on page 252.

The following example is a portion of a policy for use with the Utility Monitor.

```
<!-- ------------------------------------------------------------- -->
<!-- DEFINE THE RULES FOR THE UTILITY MONITOR                      -->
<!-- ------------------------------------------------------------- -->

    <PRACTICE NAME="STANDARDS_1">
        <UTILITY NAME="REORG_TABLESPACE">
           <MONITOR>
             <SYNTAX REMOVE="UNLOAD PAUSE"/>
             <SYNTAX VALUE="SCOPE %" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"/>
             <SYNTAX VALUE="LOG NO" SUBSTITUTE="LOG YES"/>
             <SYNTAX VALUE="SHRLEVEL %" SUBSTITUTE="SHRLEVEL NONE"/>
             <SYNTAX VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 12"/>
             <SYNTAX ADD="KEEPDICTIONARY"/>
           </MONITOR>
        </UTILITY>
        <UTILITY NAME="LOAD">
           <MONITOR>
             <SYNTAX VALUE="LOG YES" SUBSTITUTE="LOG NO"/>
             <MESSAGE ID="DSNU350I" RETURN_CODE="12" JOURNAL="NO"/>
           </MONITOR>
        </UTILITY>
        <UTILITY NAME="RUNSTATS">
            <MESSAGE ID="DSNU620I">
    </PRACTICE>
.
.
.
<DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY">
  <USE_PRACTICE NAME="STANDARDS_1"/>
    <INCLUDE>
     <RULE UTILITY_USERID="%"/>
    </INCLUDE>
</DB2SYSTEM>
.
```

Utility Monitor elements

The following elements from the example are specific to invoking the Utility Monitor and the actions that they perform.

PRACTICE
NAME="STANDARDS_1"

The name in the example is STANDARDS_1.

UTILITY
NAME="REORG_TABLESPACE"

In this example, if a REORG TABLESPACE utility is run and the subsequent syntax rules match the REORG utility, the Utility Monitor will be invoked.

MONITOR

MESSAGE
ID

RETURN_CODE
In this example, if message DSNU350I is found in the utility output, the job step that contains the LOAD utility will end with return code 12.

JOURNAL="NO"
In this example, the message text will not be written to the DB2 UET message journal table. RUNSTATS is specified here to only journal the text for message DSNU620I.

SYNTAX
REMOVE="UNLOAD PAUSE"

In this example, when the utility REORG TABLESPACE is being executed by DSNUTILB, the parameter UNLOAD PAUSE will be removed if it is present in any REORG TABLESPACE utility. If the parameter UNLOAD PAUSE is not present in the utility syntax, no action is taken.

VALUE="SCOPE %" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"
In this example, when the utility REORG TABLESPACE is being executed by DSNUTILB, the parameter SCOPE PENDING will be substituted in the utility if the parameter SCOPE is present in the utility syntax, regardless of what value is specified with SCOPE. If the parameter SCOPE is not present in the utility syntax, no action is taken. In addition, the action will not be journaled.

VALUE="LOG NO" SUBSTITUTE="LOG YES"
In this example, when the utility REORG TABLESPACE is being executed by DSNUTILB, the parameter LOG YES will be substituted into the utility if LOG NO is present. Otherwise, if the parameter LOG NO is not present, the specified value is not substituted.

VALUE="SHRLEVEL %" SUBSTITUTE="SHRLEVEL NONE"
In this example, when the utility REORG TABLESPACE is being executed by DSNUTILB, the parameter SHRLEVEL NONE will be substituted in the utility if the parameter SHRLEVEL is present in the utility syntax, regardless of what value is specified with SHRLEVEL. If SHRLEVEL is not present in the utility syntax, no action is taken.

VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 12"
In this example, when the utility REORG TABLESPACE is being executed by DSNUTILB, the parameter SORTNUM 12 will be substituted in the

utility if the parameter SORTNUM is present in the utility syntax, regardless of what value is specified with SORTNUM. If SORTNUM is not present in the utility syntax, no action is taken.

ADD="KEEPDICTIONARY"
In this example, when the utility REORG TABLESPACE is being executed by DSNUTILB, the parameter KEEPDICTIONARY will be added if the parameter is not present. If the parameter already exists, no action is taken.

DB2SYSTEM

SSID="D91A"

In this example, the SSID on which DB2 utilities are to be monitored is D91A.

ACTION="MONITOR_UTILITY"
In this example, ACTION contains a value called MONITOR_UTILITY, which will invoke the Utility Monitor for subsystem D91A.

USE_PRACTICE

NAME="STANDARDS_1"

In this example, the Practice rule named STANDARDS_1 that was previously defined in the policy will be used with the specified DB2 subsystem. The Utility Monitor will then evaluate the Practice rules defined within the policy to evaluate if the rules should be invoked.

Related concepts:

"Example policy that details RULE and RULESET sections" on page 252
In general, RULEs should be defined from the least specific to the most specific. The rules in the <RULESET> section are read and processed from top to bottom. Therefore, the order in which rules are defined is important.

"Example DSNUTILB intercept policies" on page 169
Review the following example DSNUTILB intercept policies to learn how you can create a policy to meet your intercept processing needs.

Related reference:

"<PRACTICE>" on page 142
A <PRACTICE> element enables the DSNUTILB intercept feature of DB2 UET to search the provided DB2 utility statements and perform specified actions.

"<UTILITY>" on page 156
The element <UTILITY>, when used in conjunction with the attribute NAME, identifies the DB2 utility that DB2 UET is to evaluate when DSNUTILB is invoked to verify whether the Practice rule should be enforced.

"<MONITOR>" on page 139
A element <MONITOR> identifies a set of syntax rules by which to evaluate the specified DB2 utility. This element is required when invoking the Utility Monitor and can only be specified once per <UTILITY> element.

"<SYNTAX>" on page 151
The element <SYNTAX> specifies the syntax to monitor.

"<DB2SYSTEM>" on page 131
A <DB2SYSTEM> element identifies a DB2 subsystem for which to perform DSNUTILB intercept processing to block and cancel threads or to monitor DB2 utilities. If you specify a generic wildcard pattern as its attribute value, this element can identify multiple DB2 subsystems. You must specify one or more <DB2SYSTEM> elements within the <POLICY> section. You cannot specify a <DBSYSTEM> element in a <RULESET> section.

"<USE_PRACTICE>" on page 154
A <USE_PRACTICE> element specifies the name of the predefined practice to be used when evaluating the policy rules within the element <DB2SYSTEM>, and can only be specified once.

Defining Practice rules

Defining Practice rules within the DB2 UET policy allows the Utility Monitor to scan the DSNUTILB utility syntax for comparison to the Practice rules.

Defining Practice rules by utility name is an efficient manner in which to begin using the Utility Monitor. For example, defining a utility name such as the following ensures that the Utility Monitor will only be invoked when the REORG utility is executing at the TABLEPSACE level.

```
UTILITY NAME="REORG_TABLESPACE"
```

Practice rules must be combined with RULE or RULESET within the element DB2SYSTEM to qualify when a Practice rule is invoked.

Example Practice rules

Practice rules illustrate the flexibility of the Utility Monitor.

The following examples help illustrate the control DBAs and IT Managers can employ.

Substituting a text string for another

This example focuses on when the element VALUE contains an explicit string that should be matched within the utility before the Utility Monitor is invoked.

```
<UTILITY NAME="REORG_TABLESPACE">
  <MONITOR>
   <SYNTAX VALUE="SCOPE ALL" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"/>
  </MONITOR>
</UTILITY>
```

The Utility Monitor first checks to see if the utility statement represents a REORG TABLESPACE command, as specified by the UTILITY element. If the text string SCOPE ALL is found in the syntax, then the string SCOPE PENDING is substituted in the REORG utility syntax before DSNUTILB executes.

No journal record is written to the DB2 UET Journal Tables to log the activity for this syntax rule.

Conditionally failing a utility

This example illustrates how to cause a utility to fail if a specific text string is present within utility syntax.

```
<UTILITY NAME="REORG_TABLESPACE">
  <MONITOR>
   <SYNTAX VALUE="LOG YES" FAIL="101"/>
  </MONITOR>
</UTILITY>
```

The Utility Monitor first checks to see if the utility statement represents a REORG TABLESPACE command, as specified by the UTILITY element. If it does, the Utility Monitor evaluates the second criterion. If the text string LOG YES is found in the utility syntax, then the Utility Monitor will fail

the utility, and the utility job step's return code will be that of the value specified with the FAIL attribute. In addition, as a result of the FAIL attribute, message ABPU5400E is written to the SYSPRINT data set in the job output indicating to the user why the utility job step failed.

By default, a journal record is written to the DB2 UET Journal Tables to log the activity for this syntax rule.

In addition to the presence of text strings, other criterion, such as object name or user ID, must be combined with this Practice rule within the element DB2SYSTEM to fail the utility.

In the following example, the element DB2SYSTEM contains additional criterion that will cause a utility to fail only when those criterion are met. Otherwise, DB2 UET will not fail the utility.

```
<DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY">
  <USE_PRACTICE NAME="FAIL_REORG_STANDARDS"/>
    <INCLUDE>
      <RULE UTILITY_USERID="TSOD176"/>
    </INCLUDE>
</DB2SYSTEM>
```

Because a user ID is included as additional criterion, DB2 UET will only fail the utility named in the Practice Name 'Fail Reorg Standards' if the user ID submitting the utility is TSOD176. In this manner, you can conditionally fail reorg utilities when the specified criterion is met, instead of all reorg utilities.

Unconditionally failing a utility

Utilities may be prevented from running based on any combination of elements in a rule. Some of the criterion that can be used to prevent utilities from running are object type, object name, utility command, parameters present within the utility syntax, user ID, and so on.

```
<UTILITY NAME="REORG_TABLESPACE">
    <MONITOR FAIL="1010"/>
</UTILITY>
```

Or

```
<UTILITY NAME="REORG_TABLESPACE">
    <MONITOR FAIL="1010">
    </MONITOR>
</UTILITY>
```

In this example, the utility REORG TABLESPACE will be prevented from running and a return code of 1010 will be issued by DB2 UET. By default, a journal record is written to the DB2 UET Journal Tables to log the activity for this syntax rule.

Additional criterion must then be specified within the element DB2SYSTEM to cause the Utility Monitor to be invoked and evaluate the Practice rule. Otherwise, the Utility Monitor will not fail the utility.

```
<DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY">
   <USE_PRACTICE NAME="FAIL_REORG_STANDARDS"/>
     <INCLUDE>
       <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
     </INCLUDE>
</DB2SYSTEM>
```

Because the utility command REORG TABLESPACE is included as the criterion on which to invoke the Utility Monitor, DB2 UET will only fail

the utility named in the Practice Name 'Fail Reorg Standards' if the utility being run is a REORG TABLESPACE utility. In this manner, you can unconditionally fail all REORG utilities.

Substituting a value in a keyword/value pair

This example illustrates how to substitute a value when a keyword/value pair is found within the specified syntax. In the following example, the value SORTNUM 10 will be substituted within the utility syntax if the parameter SORTNUM is found, regardless of what value is specified with SORTNUM.

```
<UTILITY NAME="REORG_TABLESPACE">
  <MONITOR>
   <SYNTAX VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 10"/>
  </MONITOR>
</UTILITY>
```

If SORTNUM 2 is specified within the utility syntax, the value SORTNUM 10 will be substituted, and the utility will execute with the parameter SORTNUM 10. If the parameter SORTNUM is not present within the utility syntax, then the value SORTNUM 10 will not be substituted within the syntax, and the utility will execute without the parameter SORTNUM.

Removing a text string from the utility syntax

This example illustrates how to remove a text string from a utility syntax if it is present.

```
<UTILITY NAME="REORG_TABLESPACE">
  <MONITOR>
   <SYNTAX REMOVE="UNLOAD PAUSE"/>
  </MONITOR>
</UTILITY>
```

The parameter UNLOAD PAUSE will be removed from the utility syntax before the utility is executed if the string value is present within the utility syntax. If the text string UNLOAD PAUSE is not present within the utility syntax, no action is taken.

Adding a text string that is not present

This example illustrates how to add a text string to the utility syntax if it is not already present.

```
<UTILITY NAME="REORG_TABLESPACE">
  <MONITOR>
   <SYNTAX ADD="KEEPDICTIONARY"/>
  </MONITOR>
</UTILITY>
```

The value KEEPDICTIONARY will be added to the utility REORG TABLESPACE if it is not already defined within the utility syntax. Otherwise, if the value KEEPDICTIONARY is already defined within the utility syntax, this rule will be ignored, and the text string will not be duplicated. This text string is added only once to the end of the utility syntax.

Ensuring a parameter is changed if it is present, or added if it is not

This example illustrates how to change a value if it is already specified in the syntax, or to add it if it is not there.

```
<UTILITY NAME="QUIESCE">
  <MONITOR>
   <SYNTAX VALUE="WRITE YES" SUBSTITUTE="WRITE NO"/>
   <SYNTAX ADD="WRITE NO"/>
  </MONITOR>
</UTILITY>
```

The string WRITE NO will be substituted into the utility syntax if WRITE YES is present. However, if the WRITE keyword is not specified in the syntax, it will be added as WRITE NO, ensuring that the QUIESCE utility is run with the required values.

Related concepts:

"Example DSNUTILB intercept policies" on page 169
Review the following example DSNUTILB intercept policies to learn how you can create a policy to meet your intercept processing needs.

"Example policy that details RULE and RULESET sections"
In general, RULEs should be defined from the least specific to the most specific. The rules in the <RULESET> section are read and processed from top to bottom. Therefore, the order in which rules are defined is important.

Example policy that details RULE and RULESET sections

In general, RULEs should be defined from the least specific to the most specific. The rules in the <RULESET> section are read and processed from top to bottom. Therefore, the order in which rules are defined is important.

In the following example policy, each set of RULEs and Practice RULEs is described in greater detail to illustrate how DB2 UET evaluates each RULE or RULESET, and how they affect utility jobs.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE OPTIONS SYSTEM "DD:DTD(ABPDTDPL)">
<!--                                                                 -->
<!-- ************************************************************** -->
<!-- *                                                            * -->
<!-- * ABPPLCY                                                    * -->
<!-- * IBM DB2 UTILITIES ENHANCEMENT TOOL FOR Z/OS V2.2 (H2AM220)  * -->
<!-- * DSNUTILB INTERCEPTION POLICY                               * -->
<!-- *                                                            * -->
<!-- ************************************************************** -->
<!-- *                                                            * -->
<!-- * 5655-T58                                                   * -->
<!-- * (C) COPYRIGHT ROCKET SOFTWARE, INC. 2002, 2016 ALL RIGHTS   * -->
<!-- * RESERVED.                                                  * -->
<!-- *                                                            * -->
<!-- ************************************************************** -->
<DSNUTILB_INTERCEPT>
<!-- ------------------------------------------------------------- -->
<!-- DEFINE THE RULES FOR DB2 OBJECTS -->
<!-- ------------------------------------------------------------- -->

<RULESET NAME="ACCOUNTS_PAYABLE">
  <INCLUDE>
    <RULE TABLESPACE="ABPQDB01.DBAP%"/>
    <RULE TABLESPACE="HREMPDB1.DBEMP%"/>
  </INCLUDE>
</RULESET>

<RULESET NAME="WEEKLY_MAINTENANCE">
  <INCLUDE>
    <RULE TABLESPACE="ABPQDB01.%"/>
    <RULE TABLESPACE="ABPQDB02.%"/>
  </INCLUDE>
```

```
</RULESET>

<RULESET NAME="APP_DEV">
  <INCLUDE>
    <RULE TABLESPACE="D91ADBB.ADTS3%" PART="1,4:10,75"/>
    <RULE TABLESPACE="D91ADBC.ADTS%"/>
  </INCLUDE>
</RULESET>

<!-- ------------------------------------------------------------ -->
<!-- DEFINE THE RULES FOR USER IDs -->
<!-- ------------------------------------------------------------ -->

<RULESET NAME="CANCEL_THESE_USERS">
  <INCLUDE>
    <RULE UTILITY_USERID="USER0%"/>
    <RULE UTILITY_USERID="USER1%"/>
    <RULE UTILITY_USERID="USER2%"/>
  </INCLUDE>
</RULESET>

<!-- ------------------------------------------------------------ -->
<!-- DEFINE THE RULES FOR SHRLEVEL -->
<!-- ------------------------------------------------------------ -->

<RULESET NAME="SHRLEVEL_RULE">
  <INCLUDE>
    <RULE SHRLEVEL="REFERENCE"/>
    <RULE SHRLEVEL="NONE"/>
  </INCLUDE>
</RULESET>

<!-- ------------------------------------------------------------ -->
<!-- DEFINE THE RULES FOR DB2 UTILITIES -->
<!-- ------------------------------------------------------------ -->

<RULESET NAME="EXPENSE_REPORTS">
  <INCLUDE>
    <RULE UTILITY_ID="UTID573%"/>
    <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
    <RULE TABLESPACE="TRVLDB.EXPRPTS"/>
  </INCLUDE>
</RULESET>

<RULESET NAME="PAYROLL">
  <INCLUDE>
    <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
    <RULE TABLESPACE="DBPR0%.%"/>
  </INCLUDE>
</RULESET>

<!-- ------------------------------------------------------------ -->
<!-- DEFINE THE RULES FOR THE UTILITY MONITOR -->
<!-- ------------------------------------------------------------ -->

<PRACTICE NAME="NORMAL_STANDARDS">
  <UTILITY NAME="REORG_TABLESPACE">
           <MONITOR>
             <SYNTAX REMOVE="UNLOAD PAUSE"/>
             <SYNTAX VALUE="SCOPE %" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"/>
             <SYNTAX VALUE="LOG NO" SUBSTITUTE="LOG YES"/>
             <SYNTAX VALUE="SHRLEVEL %" SUBSTITUTE="SHRLEVEL NONE"/>
             <SYNTAX VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 12"/>
             <SYNTAX ADD="KEEPDICTIONARY"/>
           </MONITOR>
  </UTILITY>
```

```
     <UTILITY NAME="LOAD">
       <MONITOR>
         <SYNTAX VALUE="LOG YES" SUBSTITUTE="LOG NO"/>
         <MESSAGE ID="DSNU350I" RETURN_CODE="12" JOURNAL="NO"/>
       </MONITOR>
    </UTILITY>
    <UTILITY NAME="RUNSTATS">
         <MESSAGE ID="DSNU620I">
</PRACTICE>

  <PRACTICE NAME="FAIL_REORG_STANDARDS">
    <UTILITY NAME="REORG_TABLESPACE">
      <MONITOR FAIL="100"/>
    </UTILITY>
  </PRACTICE>

<!-- ------------------------------------------------------------ -->
<!-- SPECIFY WHICH RULES TO USE ON SPECIFIC DB2 SSIDS -->
<!-- ------------------------------------------------------------ -->

<POLICY>

<!-- DB2 SSID - D91A: APPLICATION DEVELOPMENT -->

<DB2SYSTEM SSID="D91A">
  <USE_RULESET NAME="SHRLEVEL_RULE"/>
  <USE_RULESET NAME="APP_DEV"/>
</DB2SYSTEM>

<DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY">
  <USE_PRACTICE NAME="NORMAL_STANDARDS"/>
    <INCLUDE>
      <RULE UTILITY_USERID="%"/>
    </INCLUDE>
</DB2SYSTEM>

<!-- DB2 SSID - Q91A: APPLICATION TEST -->

<DB2SYSTEM SSID="Q91A" ACTION="MONITOR_UTILITY">
  <USE_PRACTICE NAME="FAIL_REORG_STANDARDS"/>
    <INCLUDE>
      <RULESET NAME="CANCEL_THESE_USERS"/>
    </INCLUDE>
</DB2SYSTEM>

</POLICY>
</DSNUTILB_INTERCEPT>
```

RULSET NAME=“ACCOUNTS_PAYABLE”

Within this RULESET named ACCOUNTS_PAYABLE, there are two table spaces being included for evaluation. Because the table space names contain wildcard characters, any match on the pattern within a DB2 utility invokes the ACTION specified in the element DB2SYSTEM.

```
<RULESET NAME="ACCOUNTS_PAYABLE">
   <INCLUDE>
     <RULE TABLESPACE="ABPQDB01.DBAP%"/>
     <RULE TABLESPACE="HREMPDB1.DBEMP%"/>
   </INCLUDE>
</RULESET>
```

In this example, if a utility is running at the table space level (such as REORG TABLESPACE) and it contains a table space name whose pattern matches ABPQDB01.DBAP% or HREMPDB1.DBEMP%, then the ACTION specified in the element DB2SYSTEM is invoked.

When two or more RULEs of the same type are included in one RULESET they are combined using the OR operator. That is, only one of these RULEs must match in order for the ACTION to be invoked.

RULESET NAME="WEEKLY_MAINTENANCE"

Within this RULESET named WEEKLY_MAINTENANCE, there are two table spaces being included for evaluation. Because the table space names contain wildcard characters, any match on the pattern found within a DB2 utility invokes the ACTION specified in the element DB2SYSTEM.

```
<RULESET NAME="WEEKLY_MAINTENANCE">
   <INCLUDE>
     <RULE TABLESPACE="ABPQDB01.%"/>
     <RULE TABLESPACE="ABPQDB02.%"/>
   </INCLUDE>
</RULESET>
```

In this example, if a utility is running at the table space level (such as REORG TABLESPACE) and it contains a table space name whose pattern matches ABPQDB01.% or ABPQDB02.%, then the ACTION specified in the element DB2SYSTEM is invoked.

When two or more RULEs of the same type are included in one RULESET they are combined using the OR operator. That is, only one of these RULEs must match in order for the ACTION to be invoked.

RULESET NAME="APP_DEV"

Within this RULESET named APP_DEV, there are two table spaces being included for evaluation. Because the table space names contain wildcard characters, any match on the pattern within a DB2 utility invokes the ACTION specified in the element DB2SYSTEM.

```
<RULESET NAME="APP_DEV">
   <INCLUDE>
     <RULE TABLESPACE="D91ADBB.ADTS3%" PART="1,4:10,75"/>
     <RULE TABLESPACE="D91ADBC.ADTS%"/>
   </INCLUDE>
</RULESET>
```

In this example, if a utility is running at the table space level (such as REORG TABLESPACE) and it contains a table space name whose pattern matches D91ADBB.ADTS3% and is executing against parts 1, parts 4 through 10, or part 75, or the pattern matches D91ADBC.ADTS%, then the ACTION specified in the element DB2SYSTEM is invoked.

When two or more RULEs of the same type are included in one RULESET they are combined using the OR operator. That is, only one of these RULEs must match in order for the ACTION to be invoked.

RULESET NAME="CANCEL_THESE_USERS"

Within this RULESET named CANCEL_THESE_USERS, there are three TSO user IDs being included for evaluation. Because the user IDs contain wildcard characters, any match on the pattern found within a DB2 utility will invoke the ACTION specified in the element DB2SYSTEM.

```
<RULESET NAME="CANCEL_THESE_USERS">
   <INCLUDE>
     <RULE UTILITY_USERID="USER0%"/>
     <RULE UTILITY_USERID="USER1%"/>
     <RULE UTILITY_USERID="USER2%"/>
   </INCLUDE>
</RULESET>
```

In this example, if a utility is submitted by a TSO user ID whose pattern matches USER0%, USER1% or USER2%, then the ACTION specified in the element DB2SYSTEM is invoked.

When two or more RULEs of the same type are included in one RULESET they are combined using the OR operator. That is, only one of these RULEs must match in order for the ACTION to be invoked.

RULESET NAME="SHRLEVEL_RULE"

Within this RULESET named SHRLEVEL_RULES, there are two SHRLEVEL values being included for evaluation. Because the SHRLEVEL values are explicitly defined and do not contain wildcard characters, the syntax in the utility must match to invoke the ACTION specified in the element DB2SYSTEM.

```
<RULESET NAME="SHRLEVEL_RULE">
   <INCLUDE>
     <RULE SHRLEVEL="REFERENCE"/>
     <RULE SHRLEVEL="NONE"/>
   </INCLUDE>
</RULESET>
```

In this example, if a utility is submitted with either SHRLEVEL REFERENCE or SHRLELVE NONE, then the ACTION specified in the element DB2SYSTEM is invoked.

When two or more RULEs of the same type are included in one RULESET they are combined using the OR operator. That is, only one of these RULEs must match in order for the ACTION to be invoked.

RULESET NAME="EXPENSE_REPORTS"

Within this RULESET named EXPENSE_RERORTS, there are three RULEs containing different values against which the utility will be evaluated.

```
<RULESET NAME="EXPENSE_REPORTS">
   <INCLUDE>
     <RULE UTILITY_ID="UTID573%"/>
     <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
     <RULE TABLESPACE="TRVLDB.EXPRPTS"/>
   </INCLUDE>
</RULESET>
```

In this example, the first RULE specifies a pattern for a DB2 utility ID. The second RULE specifies a DB2 utility command. And the third RULE specifies an object name. If a utility is submitted with a user ID that matches the DB2 utility ID pattern UTID573%, with a REORG TABLESPACE utility, and against object TRVLDB.EXPRPTS, then the ACTION specified in the element DB2SYSTEM is invoked. If any criterion doesn't match within the utility being submitted, then the ACTION defined within the element DB2SYSTEM is not invoked.

When two or more RULEs of different types are included in one RULESET they are combined using the AND operator. That is, all of the RULEs within the defined RULESET must match in order for the ACTION to be invoked.

RULESET NAME="PAYROLL"

Within this RULESET named PAYROLL, there are two RULEs containing different values against which the utility will be evaluated.

```
<RULESET NAME="PAYROLL">
   <INCLUDE>
     <RULE UTILITY_COMMAND="REORG_TABLESPACE"/>
     <RULE TABLESPACE="DBPR0%.%"/>
   </INCLUDE>
</RULESET>
```

In this example, the first RULE specifies a DB2 utility command, and the second RULE specifies a pattern for an object name. If a utility is submitted with the syntax REORG TABLESPACE and specifies a table space name matching the pattern DBPR0%.%, then the ACTION specified in the element DB2SYSTEM is invoked. If any criterion doesn't match within the utility being submitted, then the ACTION defined within the element DB2SYSTEM is not invoked.

When two or more RULEs of different types are included in one RULESET they are combined using the AND operator. That is, all of the RULEs within the defined RULESET must match in order for the ACTION to be invoked.

PRACTICE NAME="NORMAL_STANDARDS"

Within this Practice rule named NORMAL_STANDARDS, there are two utilities being monitored, each with their own set of syntax specifications. The NAME defined within the element UTILITY must match the utility submitted in order for DB2 UET to monitor the named utility. Each SYNTAX attribute will be evaluated independently, rather than evaluating all SYNTAX attributes as a group. Each syntax attribute that evaluates as true will invoke the Utility Monitor to perform the action specified by the attribute (for example, REMOVE, VALUE/ SUBSTITUTE, ADD).

```
<PRACTICE NAME="NORMAL_STANDARDS">
   <UTILITY NAME="REORG_TABLESPACE">
        <MONITOR>
         <SYNTAX REMOVE="UNLOAD PAUSE"/>
         <SYNTAX VALUE="SCOPE %" SUBSTITUTE="SCOPE PENDING" JOURNAL="NO"/>
         <SYNTAX VALUE="LOG NO" SUBSTITUTE="LOG YES"/>
         <SYNTAX VALUE="SHRLEVEL %" SUBSTITUTE="SHRLEVEL NONE"/>
         <SYNTAX VALUE="SORTNUM %" SUBSTITUTE="SORTNUM 12"/>
         <SYNTAX ADD="KEEPDICTIONARY"/>
       </MONITOR>
   </UTILITY>
```

In this example, the utility REORG TABLESPACE will be monitored for six syntax strings. The first syntax string specifies to remove the text UNLOAD PAUSE if it is present. The second syntax string specifies to substitute the given SCOPE value in the utility syntax with SCOPE PENDING. This action will not be journaled. The third syntax string specifies to substitute LOG YES instead of LOG NO if LOG NO is present. The fourth syntax string specifies to substitute the given SHRLEVEL value in the utility syntax with SHRLEVEL NONE. The fifth syntax string specifies to substitute the given SORTNUM value in the utility syntax with SORTNUM 12. The sixth syntax string KEEPDICTIONARY will be added to the utility syntax.

```
<UTILITY NAME="LOAD">
  <MONITOR>
    <SYNTAX VALUE="LOG YES" SUBSTITUTE="LOG NO"/>
    <MESSAGE ID="DSNU350I" RETURN_CODE="12" JOURNAL="NO"/>
  </MONITOR>
```

```
    </UTILITY>
    <UTILITY NAME="RUNSTATS">
         <MESSAGE ID="DSNU620I">
</PRACTICE>
```

In this example, the utility LOAD will be monitored for one syntax string. If the syntax string LOG YES is present in the utility syntax, it will be replaced by LOG NO. If message DSNU350I is found in the utility output, the job step that contains the LOAD utility will end with return code 12. The message text will not be written to the DB2 UET message journal table. RUNSTATS is specified here to only journal the text for message DSNU620I.

PRACTICE NAME="FAIL_REORG_STANDARDS"

Within this Practice rule named FAIL_REORG_STANDARDS, there is one utility being monitored. The NAME attribute of the UTILITY element specifies which utility the DB2 UET will monitor.

```
<PRACTICE NAME="FAIL_REORG_STANDARDS">
   <UTILITY NAME="REORG_TABLESPACE">
     <MONITOR FAIL="100"/>
   </UTILITY>
</PRACTICE>
```

In this example, the utility REORG TABLESPACE will not monitor syntax strings. Instead, it will simply fail the utility before it ever starts, preventing it from being run. It will also issue the return code 100 to the job step, indicating to the user that there was an error within the utility job. When this happens, a message will be issued within the SYSPRINT indicating to the submitter that the job was purposely failed and the reason why.

DB2SYSTEM SSID="D91A"

The policy definitions that were previously defined are contained within the element DB2SYSTEM. The policy rules are enforced by subsystem. The attribute SSID can contain wildcard characters, and thus apply to multiple DB2 subsystems.

```
<DB2SYSTEM SSID="D91A">
   <USE_RULESET NAME="SHRLEVEL_RULE"/>
   <USE_RULESET NAME="APP_DEV"/>
</DB2SYSTEM>
```

In this example, there is no ACTION attribute specified, so the default value of BLOCK_AND_CANCEL_THREADS will be used. Two RULESETs are included for enforcement on subsystem D91A: SHRLEVEL_RULE and APP_DEV. Each RULESET is evaluated independently. That is to say, the RULESET SHRLEVEL_RULE will be evaluated before the RULESET APP_DEV is evaluated.

The RULESET SHRLEVEL_RULE will cause threads to be blocked and canceled on the associated DB2 objects any time SHRLEVEL REFERENCE or SHRLEVEL NONE is specified. In addition, any time a utility is run on the objects associated with the RULESET APP_DEV, threads will be blocked and canceled.

DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY"

The policy definitions that were previously defined are contained within the element DB2SYSTEM. The policy rules are enforced by subsystem. The attribute SSID can contain wildcard characters, and thus apply to multiple DB2 subsystems.

```
<DB2SYSTEM SSID="D91A" ACTION="MONITOR_UTILITY">
   <USE_PRACTICE NAME="NORMAL_STANDARDS"/>
     <INCLUDE>
       <RULE UTILITY_USERID="%"/>
     </INCLUDE>
</DB2SYSTEM>
```

In this example, the ACTION attribute is specified with the value MONITOR_UTILITY, which will invoke the Utility Monitor. No threads will be canceled within this DB2SYSTEM definition; threads are canceled for this subsystem within the previously discussed DB2SYSTEM element description.

In this example, the Practice rule called NORMAL_STANDARDS will be invoked only when the RULE specifying a user ID matches the user ID value within a utility job. Because there is a wild card within the RULE UTILITY_USERID, all batch utility jobs will cause the Practice rule NORMAL_STANDARDS to be invoked.

DB2SYSTEM SSID=“Q91A” ACTION=“MONITOR_UTILITY”

The policy definitions that were previously defined are contained within the element DB2SYSTEM. The policy rules are enforced by subsystem. The attribute SSID can contain wildcard characters, and thus apply to multiple DB2 subsystems.

```
<DB2SYSTEM SSID="Q91A" ACTION="MONITOR_UTILITY">
   <USE_PRACTICE NAME="FAIL_REORG_STANDARDS"/>
     <INCLUDE>
       <RULESET NAME="CANCEL_THESE_USERS"/>
     </INCLUDE>
</DB2SYSTEM>
```

In this example, the ACTION attribute is specified with the value MONITOR_UTILITY, which will invoke the Utility Monitor. No threads will be canceled within this DB2SYSTEM definition.

In this example, Practice rule called FAIL_REORG_STANDARDS will be invoked only when the criterion contained within the RULESET CANCEL_THESE_USERS matches the user IDs associated with utility batch jobs. This PRACTICE rule allows you to purposely fail a utility if a batch job is submitted by any user ID matching those defined in the RULESET CANCEL_THESE_USERS. This function provides granularity in ensuring only those persons allowed to run a utility may do so.

Related concepts:

“Example DSNUTILB intercept policies” on page 169
Review the following example DSNUTILB intercept policies to learn how you can create a policy to meet your intercept processing needs.

“Example policy to invoke the Utility Monitor” on page 246
The policy contains rules that will be used by the Utility Monitor. The rules begin and end with the element <PRACTICE>.

“Example Practice rules” on page 249
Practice rules illustrate the flexibility of the Utility Monitor.

Related tasks:

"Creating a DSNUTILB intercept policy" on page 160
If you plan to use the DSNUTILB intercept to implement the enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads, you must define an intercept policy in XML. The policy specifies the DB2 subsystem or subsystems on which to perform intercept processing and optionally the rules for selecting the threads to block and cancel.

Journaling activity with the Utility Monitor

The actions taken by the Utility Monitor can be written to DB2 UET tables, called journaling, to track the action that is defined within the policy and the ultimate changes made to the utility syntax.

The attribute JOURNAL can be defined on either the element MONITOR or the SYNTAX child element. Because the default value on both elements is YES, omitting this attribute causes journaling to occur.

When the attribute JOURNAL="YES" is specified on the element MONITOR or is omitted entirely, all subsequent actions defined in the SYNTAX child elements are journaled, unless JOURNAL="NO" is specified on the SYNTAX child element. The JOURNAL_UTILITY table is updated for actions associated with the MONITOR element at the utility level. The JOURNAL_SYNTAX table is updated for actions associated with the SYNTAX element at the utility syntax level.

The following examples illustrate the same journaling behavior. Because JOURNAL="YES" is the default value when it is omitted, the first example causes journaling to take place for the subsequent SYNTAX child element.

```
<UTILITY NAME="QUIESCE">
 <MONITOR>
  <SYNTAX VALUE="WRITE YES" SUBSTITUTE="WRITE NO"/>
 </MONITOR>
</UTILITY>
```

```
<UTILITY NAME="QUIESCE">
 <MONITOR JOURNAL="YES">
  <SYNTAX VALUE="WRITE YES" SUBSTITUTE="WRITE NO"/>
 </MONITOR>
</UTILITY>
```

The default journaling behavior can be changed by specifying JOURNAL="NO" on either the element MONITOR or the child element SYNTAX. When JOURNAL is specified on the element MONITOR, it takes precedence over the child element SYNTAX, unless JOURNAL is specified on the element SYNTAX, overriding the specification on the parent element MONITOR.

The following syntax will not result in journaling SYNTAX activity. Because JOURNAL="NO" was specified on the parent element MONITOR, this value is used for subsequent SYNTAX child elements.

```
<UTILITY NAME="QUIESCE">
 <MONITOR JOURNAL="NO">
  <SYNTAX VALUE="WRITE YES" SUBSTITUTE="WRITE NO"/>
  <SYNTAX VALUE="EXCEPTIONS 5" SUBSTITUTE="EXCEPTIONS 0"/>
 </MONITOR>
</UTILITY>
```

The following syntax will result in journaling only those SYNTAX elements where JOURNAL="YES" is defined. The value JOURNAL="NO" was specified on the parent element MONITOR. Therefore, this value takes precedence for any

subsequent SYNTAX elements, unless JOURNAL is specified on the SYNTAX element, overriding the value on the element MONITOR.

```
<UTILITY NAME="QUIESCE">
 <MONITOR JOURNAL="NO">
  <SYNTAX VALUE="WRITE YES" SUBSTITUTE="WRITE NO" JOURNAL="YES"/>
  <SYNTAX VALUE="EXCEPTIONS 5" SUBSTITUTE="EXCEPTIONS 0"/>
 </MONITOR>
</UTILITY>
```

In this example, any time WRITE YES is found and changed to WRITE NO, then the action is journaled. However, when EXCEPTIONS 5 is changed to EXCEPTIONS 0, the action will not be journaled because JOURNAL=”NO” is specified on the element MONITOR.

How whitespace is compressed from the utility syntax

The attributes ADD, REMOVE, VALUE and SUBSTITUTE of the element SYNTAX allow administrators to dynamically change the control statements specified in the SYSIN data set.

Before the Utility Monitor can effectively search for user-defined text strings in the Practice rule (for example VALUE="LOG NO"), the utility syntax is placed into a standard internal format by removing all extraneous white space from the SYSIN. That is to say, whitespace is compressed out of the utility syntax in the SYSIN data set before the Utility Monitor searches for the text string against which to compare the policy Practice rule.

The following LOAD utility statement is an example of how whitespace in the Utility Monitor is compressed. The following example utility syntax might appear in the SYSIN data set:

```
//SYSIN DD *
LOAD DATA REPLACE LOG
   NO INTO TABLE ...
```

The DB2 UET syntax parser will “normalize” the utility syntax by removing whitespace, such that the syntax from the previous example would appear as the following, with only one space between parameters:

```
LOAD DATA REPLACE LOG NO INTO TABLE ...
```

In the following case, specifying the Practice rule with one space would match the utility syntax to invoke the Utility Monitor:

```
<SYNTAX VALUE="LOG NO" .../>
```

However, the following Practice rule with three spaces would not match the utility syntax and would not invoke the Utility Monitor:

```
<SYNTAX VALUE="LOG   NO" ... />
```

Normalizing the utility syntax before it is evaluated against policy rules makes it easier for users to define the policy rules to invoke the Utility Monitor.

Chapter 7. Writing exception rows to a data set (CHECK DATA utility enhancements)

The DB2 CHECK DATA utility with the Delete option requires an exception table for each checked table. As the rows in a table are checked, any rows that are discarded are written to the exception table. Creating and managing these exception tables is a manual task.

DB2 Utilities Enhancement Tool provides additional syntax that allows the CHECK DATA utility to write exception rows from checked tables to sequential data sets instead of DB2 tables. The discarded rows are written to a sequential data set in LOAD utility format. Additionally, LOAD control cards are written to a sequential data set as well. This allows the data to easily be loaded into another table or to be used by another application.

Topics:

DISCARDTO keywords

The CHECK DATA statement supports the DISCARDTO and optional DISCARDSPACE keywords as an alternative to the USE table_name2 clause.

The USE table_name2 clause specifies the table into which exception rows are to be copied. The DISCARDTO option is used to specify the name of a data file and a control file where discarded rows and a LOAD utility control statement will be written. The files specified in the DISCARDTO syntax can optionally be managed by TEMPLATE statements.

Restriction: You cannot use the DISCARDTO keyword with clone or auxiliary tables.

DB2 UET automatically creates a TABLESPACE and a TABLE for the rows to be discarded. The CHECK DATA utility will discard the rows to this table. DB2 UET then invokes the UNLOAD utility to unload the data to the sequential data sets that are specified by the DISCARDTO option. When the UNLOAD utility completes, the TABLE and TABLESPACE which were used for the discarded data are dropped.

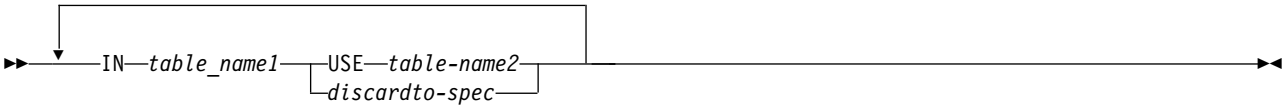
The TABLESPACE and TABLE are created in the DB2 UET work database. The name of the database in which to store work objects, the STOGROUP, and the BUFFERPOOL to be used are specified using the following options in the ABPOPTS started task initialization options member:

- WORK_DATABASE_NAME
- WORK_DATABASE_STOGROUP
- WORK_DATABASE_BUFFERPOOL

Syntax diagram for DISCARDTO

This diagram illustrates the correct syntax to use for the DISCARDTO keyword in the context of CHECK DATA FOR EXCEPTION.

CHECK DATA FOR EXCEPTION



The following diagram shows the syntax for DISCARDTO with the optional DISCARDSPACE keywords.

discardto-spec:

Descriptions of DISCARDTO operands

The operands for DISCARDTO refer to the data sets where the discarded data are written and where the LOAD control statement is written. Either name can refer to a DDNAME in the JCL, or to a TEMPLATE statement.

Operands:

unlddn

The first operand, unlddn, is required and specifies either the DDNAME or TEMPLATE name for the data set where the discarded rows will be written.

If the specified name is defined both as a DD name (in the JCL) and as a template name (in a TEMPLATE statement), it is treated as a DD name.

For more information about TEMPLATE specifications, see the chapter on TEMPLATE in the *DB2 for z/OS Utility Guide and Reference*. For more information about the UNLDDN keyword, see the chapter on UNLOAD in *DB2 for z/OS Utility Guide and Reference*.

punchddn

The second operand, punchddn, is required and specifies either the DDNAME or the TEMPLATE name for the data set where the LOAD utility control statements will be written. The LOAD control statements in this data set accurately reflect the schema of the checked tables. The table operand of the INTO TABLE specification, however, represents the discard table that was created on behalf of the DISCARDTO keyword and is no longer a valid DB2 object.

If the specified name is defined both as a DD name (in the JCL) and as a template name (in a TEMPLATE statement), it is treated as a DD name.

For more information about TEMPLATE specifications, see the chapter on TEMPLATE, in the *DB2 for z/OS Utility Guide and Reference*. For more

information about the PUNCHDDN keyword, see the chapter on UNLOAD, in the *DB2 for z/OS Utility Guide and Reference*.

data_encoding
The third operand is optional and specifies the data encoding character set to be used for the data that is discarded. EBCDIC specifies that all output data of the character type is to be in EBCDIC. ASCII specifies that all output data of the character type is to be in ASCII. UNICODE specifies that all output data of the character type (except for bit strings) is to be in Unicode. If you do not specify EBCDIC, ASCII, or UNICODE, the encoding scheme of the source data is preserved.

DISCARDSPACE (Optional) priqty and secqty
The priqty and secqty operands specify the PRIQTY and SECQTY values that will be used on the CREATE TABLESPACE statement. If you specify the DISCARDSPACE option, then you must specify both primary and secondary values. The DISCARDSPACE option must immediately follow the DISCARDTO option in the utility syntax. If you omit DISCARDSPACE and specify SPACE(priqty,secqty) on the TEMPLATE statement, then the primary and secondary values are calculated based on the space information in the TEMPLATE statement. If space information is not available from either place, the PRIQTY and SECQTY operands are omitted from the CREATE TABLESPACE statement and the TABLESPACE size values are determined by the DB2 subsystem.

The values for priqty and secqty can be -1 or from 1 through 4,194,304.

PRIQTY (PRIMARY QUANTITY) and SECQTY (SECONDARY QUANTITY) operands are used by DB2 to size underlying VSAM linear data sets when the data sets are DB2 managed.

See *SQL Reference* for discussion of PRIQTY and SECQTY operands.

Verifying that the DISCARDTO syntax is implemented

The following example output from the SYSPRINT shows this enhancement.

```
ABPU5001I 125 00:15:56.45 IBM DB2 Utilities Enhancement Tool Version 02.20, FMID=H2AM220, COMP_ID=5655-T58
ABPU5012I 125 00:15:56.45 Connected to started task ABPID=MUTA
ABPG8008I 125 00:15:56.47 System=RS25,Job=USMCD01B,Job Id=J0125870,Step=TESTCAS1,Program=DSNUTILB,User=CSTROS
ABPU5002I 125 00:15:56.48 Initialization is complete.
ABPU5004I 125 00:15:59.95 Analysis started. Step=1
ABPU5301I 125 00:16:00.20 Thread cancel prevented by policy.
ABPU5005I 125 00:16:00.21 Analysis completed. RC=0
ABPU5409I 125 00:16:00.68 SQL CREATE successful for discard table CSTROS."USMCD01.CD1_CDT_X0000001"
ABPU5411I 125 00:16:00.71 GRANT INSERT successful to discard table for authid CSTROS
ABPU5008I 125 00:16:00.71 Utility execution started. Step=1
ABPU5330I 125 00:16:00.71 Original DSNUTILB syntax follows:
ABPU5331I 125 00:16:00.71 CHECK DATA TABLESPACE ABPUSMD1.ABPUSMS1 FOR EXCEPTION IN ABPUSM.ABPUSMT1
ABPU5331I 125 00:16:00.71 DISCARDTO(DAT,PUN) DELETE YES SCOPE ALL LOG YES
ABPU5332I 125 00:16:00.71 End of original DSNUTILB syntax listing.
DSNU000I  125 00:16:00.77 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = USMCD01.CD1
DSNU1044I 125 00:16:00.80 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I  125 00:16:00.80 DSNUGUTC -  CHECK DATA TABLESPACE ABPUSMD1.ABPUSMS1 FOR EXCEPTION IN ABPUSM.ABPUSMT1
USE CSTROS."USMCD01.CD1_CDT_X0000001" DELETE YES SCOPE ALL LOG YES
```

Chapter 8. Manipulating data in input records before loading (LOAD utility enhancements)

DB2 UET provides several options for the DB2 LOAD utility to enhance load processing. These options are in addition to those that the native DB2 LOAD utility provides. They are implemented only through the use of the DB2 UET DSNUTILB intercept interface.

The options manipulate the data in the input records for the LOAD utility before the data is loaded.

Note: Do not use LOAD utility extended syntax options CONSTANT, VALUEIF, or PRESORT with tables that contain LOB columns and utilize the CONTINUEIF option.

The extended syntax options are as follows:

CONSTANT
Replaces the value for a specific field in the input records for the LOAD utility with another value that you specify. This option is useful for updating or correcting input records before the records are loaded into the database.

VALUEIF
Replaces the value for a specific field in the input records for the LOAD utility with another value that you specify. This replacement occurs only in the input records that match the "field selection criterion" that you specify in the VALUEIF statement. This option is useful for updating or correcting input records before the records are loaded into the database.

PRESORT
Sorts the rows in the input data set by table object identifier (OBID) and by clustering index key (or by the oldest defined index if no clustering index key is available), or by Hash key, before loading the data into the target tables. This option can help improve database performance.

Extended date, time, and timestamp options
Specify as input the external date, time, and timestamp formats that the IBM DB2 High Performance Unload Utility (HPU) provides as output.

IFDISCARDS
Validates records in the SYSREC file against the check constraints and data types of the table that is specified in the LOAD utility syntax.

SHRLEVEL REFERENCE
Enables support for the LOAD utility to add LOAD REPLACE SHRLEVEL REFERENCE processing.

To use these options, you will need to add them to your SYSIN for the LOAD utility. Also, you will need to create a DSNUTILB intercept policy that specifies the DB2 subsystem or subsystems on which the DSNUTILB intercept is to perform the enhanced load processing.

Topics:

- "Replacing data for a specific field (CONSTANT and VALUEIF options)" on page 269
- "Sorting rows in the input data set (PRESORT option)" on page 277
- "Specifying HPU external date, time, and timestamp formats as input" on page 278
- "Validating records before committing changes (IFDISCARDS option)" on page 281
- "Loading a table space in RO access mode (SHRLEVEL REFERENCE option)" on page 284
- "Determining whether the LOAD utility enhancements were implemented" on page 288

Related concepts:

"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

Implementing the LOAD utility enhancements

Perform the following steps to implement any of the additional options that DB2 UET supplies for the LOAD utility.

Before you begin

Before adding one of these options to a LOAD statement, review all reference and conceptual information for the option, including the proper syntax, usage considerations, and examples.

Procedure

1. In the DSNUTILB intercept policy for the DB2 UET started task that you will use for implementing the LOAD options, specify the DB2 subsystem or subsystems where you want the enhanced LOAD processing to occur. You specify a subsystem by using the <DB2SYSTEM> element within the <POLICY> section. For more information, see "Creating a DSNUTILB intercept policy" on page 160.
2. In the LOAD utility statement, add the DB2 UET option or options that you want to use. Make sure that you use the proper syntax.
3. Ensure that the DSNUTILB intercept status is enabled. You can display the intercept status by issuing the DISPLAY INTERCEPT command from the z/OS console. If the intercept is disabled, you will need to activate it by using the ACTIVATE INTERCEPT console command. For more information, see "Console commands for the started task" on page 447.

Results

When DSNUTILB invokes the LOAD utility, the option or options that you added will be used to perform the enhanced load processing.

What to do next

You can check the LOAD output to determine if the specified LOAD options were implemented. See “Determining whether the LOAD utility enhancements were implemented” on page 288.

Tip: DB2 UET uses a default value of 20 for the BUFNO parameter to set the number of buffers for the input data set for the LOAD utility. If you notice LOAD performance degradation, you can specify a higher value for the BUFNO parameter in the INDDN data definition (DD) statement (SYSREC by default) of the LOAD JCL or in the TEMPLATE setup statement for the utility. Refer to the *DB2 Version 9.1 for z/OS Utility Guide and Reference* for more information.

Restriction: When the LOAD utility is used in conjunction with the PART keyword, the extended syntax that is implemented by DB2 UET is not supported. In the following example, note the presence of the PART keyword and the extended syntax keywords of PRESORT, CONSTANT and VALUEIF. The extended syntax is ignored when the PART keyword is specified.

```
LOAD DATA REPLACE LOG YES
 PRESORT
 INTO TABLE TB_CREATOR.TB_NAME
   PART 001 INDDN SYSREC01
   (
   COL_1 POSITION (6:13) CHAR CONSTANT VALUE = 'XXXXXXXX' ,
   COL_2 POSITION (16:23) CHAR ,
   COL_3 POSITION (39:20) CHAR VALUEIF COL3='DAVIS' VALUE='XXXXX'
   )
```

Related concepts:

“Replacing data for a specific field (CONSTANT and VALUEIF options)”
The CONSTANT and VALUEIF options both replace data for a specific field in the input records for a table before the LOAD utility loads the records.

“Determining whether the LOAD utility enhancements were implemented” on page 288
To determine whether the DB2 UET options for the LOAD utility were processed, check the DB2 UET messages in the SYSPRINT data set for the LOAD utility. Use SDSF or an equivalent tool to view this information.

“Defining and using a DSNUTILB intercept policy” on page 129
If you want to use the DSNUTILB intercept component to implement the DB2 UET enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads automatically for certain DB2 utility operations, you must define a DSNUTILB intercept policy.

Related reference:

“Sorting rows in the input data set (PRESORT option)” on page 277
Use the PRESORT option for the DB2 LOAD utility to sort the rows in the input data set before loading the data into the target tables. DB2 UET invokes DB2 Sort for z/OS to provide sort processing.

Replacing data for a specific field (CONSTANT and VALUEIF options)

The CONSTANT and VALUEIF options both replace data for a specific field in the input records for a table before the LOAD utility loads the records.

The CONSTANT option replaces the data in a particular field across all input records for a table, whereas the VALUEIF option replaces the data in a particular field only in the input records that meet a "field selection criterion" (field-value

criterion) that you specify. To learn more about these options, review the syntax diagram, the topic on each option, and the topics on padding and truncation of replacement values.

Note: The input data set remains unchanged, but the specific column in the table is populated with the replaced data value as dictated by the CONSTANT or VALUEIF keyword.

Related tasks:

"Implementing the LOAD utility enhancements" on page 268
Perform the following steps to implement any of the additional options that DB2 UET supplies for the LOAD utility.

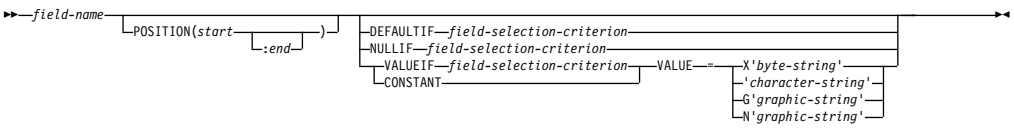
Syntax diagram for the CONSTANT and VALUEIF options

This syntax diagram illustrates the correct syntax to use for adding the DB2 UET CONSTANT and VALUEIF options to a DB2 LOAD utility statement.

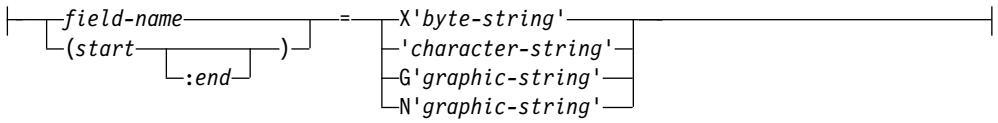
The options are specified in the *field specification* portion of an INTO-TABLE specification for the LOAD utility and pertain to a specific field in a table to be loaded. The CONSTANT and VALUEIF options cannot be specified together or with the DEFAULTIF or NULLIF option in the same field specification.

The *field selection criterion* is either a field name or byte position plus a literal value for that field. For detailed information about the correct syntax for the field selection criterion, see the information for the WHEN clause in the LOAD utility section of the IBM *DB2 for z/OS Utility Guide and Reference*.

CONSTANT and VALUEIF options in a field specification



Field selection criterion:



CONSTANT option

Use the CONSTANT option for the DB2 LOAD utility to replace existing data in a particular field across all input records for a table prior to loading the records into the table.

The CONSTANT option is specified in the *field specification* portion of an INTO-TABLE specification within the LOAD utility syntax. The option pertains to a specific column in a table to be loaded. When correctly specified, this option enables DB2 UET to replace the data in the target column with a replacement value that you specify *before* the data is loaded.

Note: The input data set remains unchanged, but the specific column in the table is populated with the replaced data value as dictated by the CONSTANT keyword.

General syntax

Use the following general syntax to add the CONSTANT option to a LOAD field specification:

```
"target-field" POSITION(position) data_type CONSTANT VALUE = 'replacement_value'
```

Where:

- *target-field* is the name of the field in the input records in which you want to replace existing values with another value. (required)
- POSITION is the option that specifies the byte position of the target field. The *byte_position* can be a *start*:*end* byte specification, a *start* byte only, or a relative byte position. (optional)
- *data_type* is the data type of the target field. It can include a length designation, for example, CHAR(6). (optional)
- CONSTANT is the name of the subject option. (required)
- VALUE is the option that specifies the replacement value (the value that will replace the existing data in the target field in the input records). (required)

Usage considerations

Consider the following usage issues:

- If the LOAD field specification identifies a target field that does not exist in the target table and the IGNOREFIELDS YES option is specified for the table, the CONSTANT statement for that target field will be ignored.
- When using the CONSTANT option, you cannot specify either `ABPREC` or `ABP@@@` as the INDDN DD name or template name for the LOAD utility. You also cannot specify these values in any DD statement in the entire DSNUTILB job that includes the LOAD operation.

Example 1

Assume that the following CONSTANT statement is specified in an INTO-TABLE specification of a LOAD job:

```
"PHONE" POSITION(10:24) CHAR CONSTANT VALUE = '123-456-7890'
```

This example statement replaces the existing data in the PHONE field at byte positions 10 through 24 with the replacement phone number of 123-456-7890. The target field length of 15 bytes, as determined from the POSITION *start*:*end* byte values, is long enough to accommodate the entire replacement value of 12 bytes. When the value is inserted into the target field, it will be padded on the right with blanks to fill out the 15-byte length.

Example 2

Assume that the following CONSTANT statement is specified in an INTO-TABLE specification of a LOAD job:

```
"ADDRESS" POSITION(30) CHAR(20) CONSTANT VALUE = '1234 MY STREET, SUITE 567"
```

This example statement replaces the existing data in the ADDRESS field of the input records with a replacement address. The ADDRESS field begins at byte 30 in the input records (as indicated by the POSITION option) and is 20 bytes in length (as indicated by the CHAR(20) data type). Because the replacement address length (25 bytes) is longer that the target field length (20 bytes), the replacement value will be truncated to 1234 MY STREET, SUIT.

VALUEIF option

Use the VALUEIF option for DB2 LOAD utility to replace the data at a particular position in the input records for a table prior to loading the records into the table. This replacement occurs only in the input records that meet the "field selection criterion" that you specify.

The VALUEIF option is specified in the *field specification* portion of an INTO-TABLE specification within the LOAD utility syntax. The option pertains to a specific column in a table to be loaded.

Note: The input data set remains unchanged, but the specific column in the table is populated with the replaced data value as dictated by the VALUEIF keyword

The VALUEIF statement includes a *field selection criterion* that is matched against the input records for the table to determine if the replacement should occur. The field selection criterion is composed of a field name or byte position plus a literal value for that field. The literal value is called the *comparison string*.

DB2 UET compares the field selection criterion to the actual data in the input records for the table. If a match is found, DB2 UET replaces the data in the target field with the replacement value that you specify *before* the data is loaded.

General syntax

Use the following general syntax to add the VALUEIF option to a LOAD field specification:

```
"target_field" POSITION(byte_position) data_type VALUEIF field_selection_criterion
VALUE = 'replacement_value'
```

Where:

- *target-field* is the name of the field in the input records in which you want to replace existing values with another value. (required)
- POSITION is the option that specifies the byte position of the target field. The *byte_position* can be a *start*:*end* byte specification, a *start* byte only, or a relative byte position. (optional)
- *data_type* is the data type of the target field. It can include a length designation, for example, CHAR(6). (optional)
- VALUEIF is the name of the subject option. (required)
- *field_selection_criterion* is a field name or byte position plus a literal value for that field, which is matched against the input records to determine the input records in which the data in the target field should be replaced by the specified replacement value. (required)
- VALUE is the option that specifies the replacement value (the value that will replace the existing data in the target field if the field selection criterion matches the input records). "VALUE" must be followed by the equals (=) operator and then the replacement string enclosed in single quotation marks. (required)

Comparison of the field selection criterion to input records

To determine whether the data in the target field should be replaced by the replacement value, DB2 UET compares 1) the literal value that is specified in the field selection criterion (the comparison string) to 2) the actual data in the input records for the field that is specified by field name or byte position in the field selection criterion (the comparison field). This comparison occurs before the input

records are loaded. If a match is found for an input record, the replacement value will be loaded into the target field in that input record. If a match is not found, the replacement will not occur.

When making the comparison, DB2 UET interprets the comparison string characters as EBCDIC. If the input records contain ASCII or UNICODE data for the comparison field, you should specify the comparison string in hexadecimal format. If the length of the comparison string does not match the length of the data in the comparison field, DB2 UET will pad the shorter of the two with blanks (for character or graphic data) or with zeroes (for hexadecimal data) before making the comparison.

When determining whether truncation of the comparison string is necessary, DB2 UET compares the length of the comparison string to one of the following lengths, depending on whether you specified a field name or byte position in the field selection criterion:

- If you specified a *start*:*end* byte position in the field selection criterion, the comparison string length is compared to the length that is based on that byte position, that is, (*end* - *start*) + 1.
- If you specified a *start* byte position without an *end* byte position, the comparison string length is compared to its own length. Therefore, no truncation will occur.
- If you specified a *field name*, the comparison string length is compared to the length of the target field. The length of the target field is one of the following (in order of priority): 1) the length that is based on the POSITION value for the target field, 2) the length that is specified in the data type for the target field, or 3) the length that is specified in the DB2 catalog for the column corresponding to the target field.

For example, assume that the field selection criterion is `VALUEIF (STATE) = 'CALIFORNIA'`. Because a field name (STATE) rather than a byte position is specified in the field selection criterion, DB2 UET compares the length of the comparison string 'CALIFORNIA' (10 bytes) to the target field length. Assume that the target field length is 2 bytes based on its data type of CHAR(2). Because the comparison string is longer than this target field length, the string will be truncated to 'CA' before it is compared to the actual data for the field STATE in the input records.

Usage considerations

Consider the following usage issues:

- If the input data set for the LOAD utility is in a delimited format (as indicated by the DELIMITED option), you cannot specify a *start*:*end* byte position in the field selection criterion for the VALUEIF option.
- If the target field has the data type of BINARY, and if the replacement value is specified in hexadecimal format and is shorter than the target field length, DB2 UET will pad the replacement value on the right with zeroes. This type of padding might be inappropriate for some data types such as integers. For such data types, you should manually pad the replacement value on the left with leading binary zeroes.
- If the LOAD field specification identifies a target field that does not exist in the target table and the IGNOREFIELDS YES option is specified for the table, the VALUEIF statement for that target field will be ignored.

- When using the VALUEIF option, you cannot specify either ABPREC or ABP@@@ as the INDDN DD name or template name for the LOAD utility. You also cannot specify these values in any DD statement in the entire DSNUTILB job that includes the LOAD operation.
- If you want to specify a replacement value in the VALUE= clause that is not one of the supported types of character or graphic strings, you must specify the value in hexadecimal format. (The graphic strings are for double-byte characters.) For more information, see "Syntax diagram for the CONSTANT and VALUEIF options" on page 270. For example, if you want to specify a decimal number or integer as the replacement value, you would need to do so in hexadecimal format.

Example 1

Assume that the following VALUEIF statement is specified in an INTO-TABLE specification of a LOAD job:

```
"SUPPLIER" POSITION(1:5) CHAR VALUEIF (15:16) = 'CA' VALUE = 'ABCCO'
```

With this example statement, the comparison string CA is matched against the data at the byte positions 15 through 16 in the input records for the LOAD utility. If a match is found in any input record, the replacement value ABCCO replaces the existing data in the SUPPLIER field in that record before the record is loaded. Note that the comparison string CA does not need to be truncated because the comparison string length and the length based on the 15:16 byte position are the same (2 bytes). Also, the replacement value does not need to be padded or truncated because its length exactly matches that of the SUPPLIER field based on the field position.

Example 2

Assume that the following VALUEIF statement is specified in an INTO-TABLE specification of a LOAD job:

```
"SHIPPER" POSITION(20) CHAR(2) VALUEIF (STATE) = 'WASHINGTON' VALUE = 'XYZSHIPMENTS'
```

With this example statement, the comparison string WASHINGTON is matched against the data for the STATE field in the input records for the LOAD utility. If a match is found in any input record, the truncated replacement value XY replaces the existing data in the SHIPPER field in that record before the record is loaded. The truncation of the replacement value is necessary because XYZSHIPMENTS (12 bytes) is longer than the target field length, as determined by the data type CHAR(2). Note that the comparison string WASHINGTON will also be truncated to WA before it is matched against the data in the STATE field because it is longer than the target field length of 2.

Example 3

Assume that the following VALUEIF statement is specified in an INTO-TABLE specification of a LOAD job:

```
"PMARGIN" POSITION(15:18) INTEGER(4) VALUEIF (REGION) = 'WEST' VALUE = X'0007003F'
```

With this example statement, the comparison string WEST is matched against the data for the REGION field in the input records for the LOAD utility. If a match is found in any input record, the hexadecimal replacement value X'0007003F' replaces the existing data in the PMARGIN field in that record before the record is loaded. Because the data type of the target field is INTEGER, the replacement value must

be specified in hexadecimal format. The replacement value does not need to be padded or truncated because its length exactly matches that of the PMARGIN field based on the POSITION byte range.

Padding of CONSTANT and VALUEIF replacement values

If the value that is specified in the VALUE option of a CONSTANT or VALUEIF statement is shorter than the fixed length of the field in which it is to be inserted, DB2 UET will "pad" the field on the right with zeroes or blanks.

The padding character is a binary zero if you specify the replacement value in the CONSTANT or VALUEIF statement in hexadecimal format. Otherwise, the padding character is a blank. The representation of a blank depends on the data type of the target field and on the coded character set identifier (CCSID) of the target table. For example, if the character set is UTF-8, the padding character is x20.

Padding occurs only for fixed-length fields that have the following data types:

- CHAR
- GRAPHIC
- BINARY

To determine if padding is necessary, DB2 UET compares the length (number of bytes) of the replacement value that is specified by the VALUE option in the CONSTANT or VALUEIF statement to one of the following lengths (in order of priority) for the target field:

- If the POSITION option is specified for the target field and includes *start*:*end* byte values, the length is calculated as follows: (*end* - *start*) + 1. For example, if POSITION(1:3) is specified, the replacement value length is compared to the target field length of 3.
- If the data type of the target field includes a length value, that length is used, provided that no POSITION(*start*:*end*) values are also specified. For example, if the data type is CHAR(6), the replacement value length is compared to the target field length of 6 from the data type specification.
- If no field length is provided by the POSITION option or the data type, the length that is recorded in the DB2 catalog for the column corresponding to the target field is used.

For variable-length fields that have the data types VARCHAR, VARGRAPHIC, and VARBINARY, padding does not occur. The field length in the input record is changed to the length of the value that is specified in the VALUE option of the CONSTANT or VALUEIF statement. No padding characters are needed because the length of the value to be loaded matches the field length.

Example 1

Assume that the following CONSTANT statement is specified in an INTO-TABLE specification of a LOAD job:

```
"FIELDA" POSITION(1:6) CHAR(6) CONSTANT VALUE = 'ABC'
```

Because the length of the replacement value 'ABC' (3 bytes) is less than the fixed length of the target field (6 bytes), as indicated by the POSITION *start*:*end* byte values, padding will occur. DB2 UET will add three blanks (the padding

characters) after ABC and load this entire value at byte position 1:6 in the input records.

Example 2

Assume that the following VALUEIF statement is specified in an INTO-TABLE specification of a LOAD job:

```
"FIELDB" POSITION(7) VARCHAR(20) VALUEIF (1:4) = 'TB1A' VALUE = 'DEF'
```

The length of the value 'ABC' (3 bytes) is less than the existing length of the VARCHAR target field. However, padding will not occur. The length of the target field will be changed to 3 bytes, and the value DEF (without any padding characters) will replace the data in this field. The replacement will occur only in input records that include the value 'TB1A' at byte position 1:4.

Truncation of CONSTANT and VALUEIF replacement values

If the replacement value that is specified in a CONSTANT or VALUEIF statement is longer than the length of the target field in which it is to be inserted, DB2 UET will truncate the replacement value.

Truncation occurs for fixed-length fields that have the following data types only:

- CHAR
- GRAPHIC
- BINARY

To determine if truncation is necessary, DB2 UET compares the length (number of bytes) of the replacement value that is specified by the VALUE option in the CONSTANT or VALUEIF statement to one of the following lengths (in order of priority) for the target field:

- If the POSITION option is specified for the target field and includes *start*:*end* byte values, the length is calculated as follows: (*end* - *start*) + 1. For example, if POSITION(1:3) is specified, the replacement value length is compared to the target field length of 3.
- If the data type of the target field includes a length value, that length is used, provided that no POSITION(*start*:*end*) values are also specified. For example, if the data type is CHAR(6), the replacement value length is compared to the target field length of 6 from the data type specification.
- If no field length is provided by the POSITION option or the data type, the length that is recorded in the DB2 catalog for the column corresponding to the target field is used.

Truncation might also occur for variable-length fields that have the data types VARCHAR, VARGRAPHIC, and VARBINARY. If the longest length of a variable-length field cannot accommodate the replacement value, the replacement value will be truncated to the maximum length that is specified in the catalog for the target field.

Example 1

Assume that the following CONSTANT statement is specified in an INTO-TABLE specification of a LOAD job:

```
"FIELDA" POSITION(1:6) CHAR(6) CONSTANT VALUE = 'ABCDEFGHIJKLMN'
```

Because the length of the replacement value 'ABCDEFGHIJKLMN' (14 bytes) is longer than the fixed length of the target field, as indicated by the POSITION(1:6) option, truncation will occur. The truncated string 'ABCDEF' will replace the data at byte position 1:6 in the input record.

Example 2

Assume that the following VALUEIF statement is specified in an INTO-TABLE specification of a LOAD job and the VALUEIF field selection criterion (that is, `VALUEIF (10:15) = 'XYZ999'`) matches one or more records in the table that contains the target field:

```
"FIELDB" POSITION(1) CHAR(3) VALUEIF (10:15) = 'XYZ999' VALUE='ABCDEFGHIJKLMN'
```

DB2 UET compares the length of the replacement value 'ABCDEFGHIJKLMN' (14 bytes) to the target field length that is calculated based on the length that is specified in the CHAR(3) data type (3 bytes). Because the 14-byte replacement value is longer than the 3-byte field length, truncation will occur. The truncated string 'ABC' will replace the data at byte position 1:3 in the input record.

Sorting rows in the input data set (PRESORT option)

Use the PRESORT option for the DB2 LOAD utility to sort the rows in the input data set before loading the data into the target tables. DB2 UET invokes DB2 Sort for z/OS to provide sort processing.

Syntax

The PRESORT option is specified for the entire LOAD statement as shown in the following syntax diagram. (This syntax diagram is a fragment of the LOAD syntax diagram and does not show other native LOAD options that can be specified at this level.)

Sort criteria

DB2 UET determines which sort criteria to use based on the type of table space to be loaded, as indicated in the following table:

Table 28. PRESORT sort criteria by table space type

Table space/Table type	Sort rows by
Partitioned or universal	By clustering index key, or if no clustering index key exists, by the oldest defined index if available.
Simple or segmented	By table object identifier (OBID) first and then by clustering index key. If no clustering index key is available, sort rows by the oldest defined index instead.
ORGANIZE BY HASH	Hash key

If a sort criterion is not defined and no other valid sort criterion is available for a table space, the rows to be loaded into the tables in that table space will not be presorted. For example, if no clustering index key or index is defined for a universal table space, DB2 UET does not presort rows prior to loading them into

tables in that table space. A table defined as ORGANIZE BY HASH is always sorted by the hash key.

Usage considerations

Consider the following usage issues:

- When the PRESORT option is specified, DB2 UET uses the SORTKEYS or NUMRECS, SORTNUM, and SORTDEVT options according to rules that are specified in the *DB2 for z/OS Utility Guide and Reference*, Version 9.1 or later. The SORTKEYS value should reflect the number of records in SYSREC. DB2 UET recalculates a new value for SORTKEYS and passes it to DSNUTILB. If you specified 0 or an integer as the SORTKEYS value, your specified value will be replaced by the value that DB2 UET calculates. However, if you specified NO as the SORTKEYS value, this replacement will not occur. In this case, the SORTKEYS option will not be used. For the SORTDEVT and SORTNUM options, DB2 UET simply passes any value that you specified for these options to the SORT program for the LOAD utility for use in PRESORT processing.
- If you are loading a large number of rows and performing discard processing, the PRESORT option might cause the performance of the LOAD utility to be degraded. DB2 UET sorts the data in the input data set twice:
  - When providing the data to the LOAD utility for load processing.
  - When providing the data to the LOAD utility for discard processing.

  In this situation, sort the data in the input data set before running the LOAD utility, rather than use the PRESORT option.
- DB2 UET does not sort data for fields that have a large object (LOB) data type such as BLOB, CLOB, or DBCLOB.
- DB2 UET does not sort data by an index key (a clustering index key, or if no clustering index is available, the key of the oldest defined index) if the key includes one or more columns with one of the following unsupported data types: DECFLOAT, DISTINCT, XML.
- When using the PRESORT option, you cannot specify either `ABPREC` or `ABP@@@` as the INDDN DD name or template name for the LOAD utility. You also cannot specify these values in any DD statement in the entire DSNUTILB job that includes the LOAD operation.

Note: The following restrictions apply:

- You must include a LOAD field specification for all columns that are defined as a key column of the index that will be used by the PRESORT option.
- A failure in the DB2 UET parser will now occur (including the informative message ABPP9939E) when the PRESORT option is used with the FORMAT UNLOAD, FORMAT SQLDS, FORMAT INTERNAL keywords and in those instances where no field specifications are supplied with the LOAD utility. DB2 UET will terminate the utility before the utility is executed by DSNUTILB.

Related tasks:

"Implementing the LOAD utility enhancements" on page 268
Perform the following steps to implement any of the additional options that DB2 UET supplies for the LOAD utility.

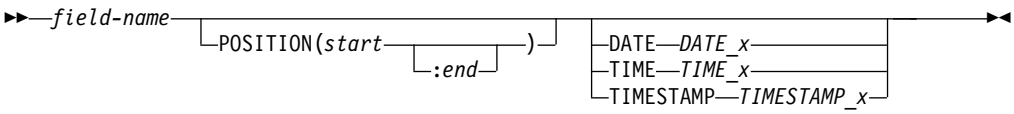
Specifying HPU external date, time, and timestamp formats as input

Specify as input the external date, time, and timestamp formats that the IBM DB2 High Performance Unload Utility (HPU) provides as output. The HPU product is not required; DB2 UET simply converts HPU formats for use by the LOAD utility.

Syntax diagram for date, time, and timestamp options

This syntax diagram illustrates the correct syntax to use for adding the date, time, and timestamp options to a DB2 LOAD utility statement. The options are specified in the *field specification* portion of an INTO-TABLE specification for the LOAD utility and pertain to a specific field in a table to be loaded.

DATE, TIME, and TIMESTAMP options in a field specification



To specify the extended formats, in an INTO-TABLE specification within the LOAD utility syntax, replace the DATE EXTERNAL and TIME EXTERNAL option values with a date, time, or timestamp value that is shown in Table 29. For more information, see the examples that follow the table.

Table 29. Extended date/time/timestamp formats for LOAD utility

Date/time/timestamp values	Format
DATE_A	MM-DD-YYYY
DATE_B	MM-DD-YY
DATE_C	YYYY-MM-DD
DATE_D	YY-MM-DD
DATE_E	DD-MM-YYYY
DATE_F	DD-MM-YY
DATE_G	YYYY-DDD
DATE_H	YY-DDD
DATE_I	MMDDYYYY
DATE_J	MMDDYY
DATE_K	YYYYMMDD
DATE_L	YYMMDD
DATE_M	DDMMYYYY
DATE_N	DDMMYY
DATE_O	YYYYDDD
DATE_P	YYDDD
TIME_A	HH.MM.SS
TIME_B	HH.MM
TIME_C	HH.MM AM or PM
TIME_D	HHMMSS
TIME_E	HHMM
TIMESTAMP_A	YYYY-MM-DD-HH.MM.SS
TIMESTAMP_C	YYYYMMDDHHMMSS
TIMESTAMP_D	YYMMDDHHMMSS

Restrictions and considerations

- This enhancement only supports padded records with the POSITION parameter defined in the field definitions.
- Variable-length timestamps are not supported (TIMESTAMP_B, TIMESTAMP_E, TIMESTAMP_F).
- Before the LOAD utility executes, DB2 UET converts the extended LOAD syntax to the following formats:
  - All dates are converted to DB2 ISO format YYYY-MM-DD.
  - HH:MM:SS (PM time is converted to military time.)
  - HH.MM PM (Military time cannot be entered because it converts to military time.)
- Two-digit years (that is, MM-DD-YY) are expanded to four digits using 100 Year Fixed Window, 1930-2029.
- The valid Julian date range is 1 through 365 (366 for leap years).
- Julian date conversion uses packed decimal operations.
- Conversions are performed after CONSTANT and VALUEIF substitution. Values that you specify for substitution using the CONSTANT and VALUEIF options must match exactly the original data in the record.

  For example, the correct format for TIME_C is HH.MM PM (or AM). The following example shows the correct format for the VALUEIF value. Any other TIME format for the VALUEIF value in this example would be incorrect.

```
     , "TIMEC1"
 POSITION(  00047:00054) TIME_C
                         CONSTANT VALUE='10.20 AM'
, "TIMEC2"
 POSITION(  00056:00063) TIME_C
   VALUEIF TIMEC2='13.59 PM' VALUE='12.59 PM'
```

- DEFAULTIF criteria on which to match must be specified in ISO format. If you specify "COLUMN NAME" (instead of the exact position) for SELECTION_CRITERIA, you must specify the entire value for comparison, including all trailing zeroes for the timestamp.
- DB2 UET performs data conversion for the HPU date, time, and timestamp formats, and constructs the resulting record for each HPU-format field as follows:
  - If the specified value is shorter than the length of the data in ISO format, then the converted value in ISO format is placed after the original record.
  - If the specified value is longer than or equal to the length of the data in ISO format, then the converted value in ISO format replaces the original record.
- DB2 UET performs data validation for the HPU Julian dates and AM/PM time formats. During validation, when DB2 UET encounters a Julian date that is not valid, it pads the value with binary nulls (x'00') up to 10 bytes to force DB2 to discard that record. The specified value is not changed and appears in the discard data set. When DB2 UET encounters a TIME_C value that is not valid, it leaves the value in the same position in the original record (because lengths are equal) so that DB2 will discard the record as not valid. In TIME_C format, when AM or PM is not qualified, DB2 cannot correctly recognize an invalid value. Therefore, is this case, DB2 UET inserts one binary null (x'00') between hours and minutes in the original record. This action forces DB2 to discard the record, but retains a recognizable value in the discarded record.

Example 1: LOAD basic syntax

The following example shows a standard LOAD using basic syntax:

```
LOAD DATA INDDN SYSREC REPLACE
  SORTDEVT SYSDA
  DISCARDDN DISC1
  ERRDDN    ABPERR
  MAPDDN    ABPMAP
  LOG YES
  INTO TABLE BACTEST.ABPPBR_TB1
       (
       "NAME1"
       POSITION(  00004:00004) CHAR(00001)
       , "NAME2"
        POSITION(  00006:00017) VARCHAR
       , "NAME3"
        POSITION(  00019:00028) DATE EXTERNAL
       , "NAME4"
        POSITION(  00079:00082) TIME EXTERNAL
       , "NAME5"
         POSITION(  00226:00244) TIMESTAMP EXTERNAL
       )
```

Example 2: LOAD extended syntax

The following example shows LOAD syntax using extended formats:

```
LOAD DATA INDDN SYSREC REPLACE
  SORTDEVT SYSDA
  DISCARDDN DISC1
  ERRDDN    ABPERR
  MAPDDN    ABPMAP
  LOG YES
  INTO TABLE BACTEST.ABPPBR_TB1
       (
       "NAME1"
        POSITION(  00004:00004) CHAR(00001)
       , "NAME2"
        POSITION(  00006:00017) VARCHAR
       , "NAME3"
        POSITION(  00019:00028) DATE_A
       , "NAME4"
        POSITION(  00079:00082) TIME_A
       , "NAME5"
        POSITION(  00226:00244) TIMESTAMP_A
       )
```

Validating records before committing changes (IFDISCARDS option)

The IFDISCARDS option for the DB2 LOAD utility provides Load Prevalidate feature.

When you specify the IFDISCARDS extended syntax option, before committing changes to the database, DB2 UET validates records in the SYSREC file against the check constraints and data types of the table that is specified in the LOAD utility syntax. If it detects issues, you can review them before performing the load, correct the errors, and rerun the job without performing a recovery of the objects.

The IFDISCARDS option indicates to the DB2 UET DSNUTILB intercept program that you want to use the Load Prevalidate feature and the action that you want taken when rows are flagged for discard. Rows found to be in violation are written to the data set that is specified by the DISCARDDN option.

When the LOAD utility job includes the IFDISCARDS option, processing is as follows:

1. DB2 UET creates a new set of shadow objects against which to perform the load.

   Shadow objects are duplicates that mirror the original database, table spaces, tables, and indexes. Shadow tables are created with same OBID as the original. DB2 UET creates a unique name for the shadow objects by using the prefix that is specified in the option **Shadow database prefix** and adding a numeric suffix. In the shadow database, DB2 UET creates shadow copies of the table space, table, and all indexes on the table that is being loaded, and uses the value of the **Shadow schema** option as the schema name for the objects.
2. DB2 UET copies check constraints from production to shadow objects and determines the data set list of the objects that are being loaded.
3. DB2 UET places the production (original) objects in read-only access mode during processing.
4. If you specify RESUME YES, the contents of the existing production data sets are copied to shadow data sets (FlashCopy® is used, if available). After copying the data DB2 UET runs a repair utility on the shadow objects to make the OBID, version, and level information of the copied data sets match the production objects.

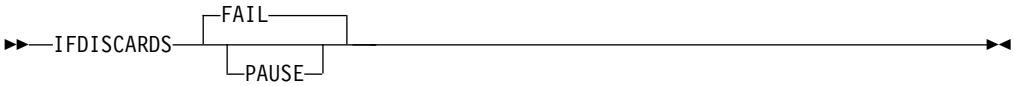
   During this process, DB2 UET places the production object in read-only (RO) mode and briefly stops the shadow objects (allowing no access).
5. If an inline image copy was requested, the image copy is removed from the LOAD utility syntax.
6. DB2 UET loads the data into the shadow table instead of the production table.

   During the load, the *production* table is available for select processing.
7. After a successful LOAD utility job, processing is as follows:
   a. DB2 UET examines the shadow objects and repairs identifiers as necessary to match production objects.
   b. If an inline copy was requested and removed from the syntax, DB2 UET invokes the COPY utility to take a full image copy after the LOAD utility completes.
   c. A brief outage begins when DB2 UET stops the production spaces.
   d. To make the loaded data accessible in the new production object, DB2 UET swaps the underlying VSAM data sets for the production and shadow objects.
   e. DB2 UET updates real-time statistics (RTS) for the production table.
   f. If referential constraints on the old production table were detected, DB2 UET sets CHECK-pending (CHKP) status on the new production object, if appropriate.
   g. DB2 UET starts the new production object in the state that it was in before the load, and the outage ends.
   h. DB2 UET deletes renamed temporary data sets and drops the old production objects.

Note: If DB2 UET encounters errors, it responds as determined by the IFDISCARDS option value.

Syntax for the IFDISCARDS option

The IFDISCARDS option is specified for the entire LOAD statement as shown in the following syntax diagram. (This syntax diagram is a fragment of the LOAD syntax diagram and does not show other native LOAD options that can be specified at this level.)



If the IFDISCARDS option is specified and no operand is present, FAIL is the default value. If the option IFDISCARDS is not present in the LOAD utility syntax, prevalidation does not occur, and the LOAD utility processes the table as usual.

FAIL IFDISCARDS FAIL (the default) tells DB2 UET to discontinue the LOAD utility if data records are discarded, then delete the shadow objects, and leave the production table unaltered in the same state as before the load started. The utility and its corresponding utility ID are terminated automatically and cannot be restarted. The return code is set to 8. The table data is untouched by the load prevalidation process.

PAUSE

IFDISCARDS PAUSE tells DB2 UET to pause the LOAD utility after performing load prevalidation processing if data records are discarded. You can review the data rows that were flagged for discard and decide whether to restart the LOAD utility and discard the data rows, or terminate the LOAD utility, which leaves the production table data untouched.

The LOAD utility ID is not terminated, allowing the LOAD utility job to be restarted. The return code is set to 4. The job will not appear in the report generated by the **–DIS UTIL** command. That is, the job is not in a stopped state because DB2 has no record of it in the SYSUTIL directory table.

The job is recorded in the DB2 UET worklist table as paused, and you can restart it by rerunning the job with the same utility ID. The LOAD utility processes the table by discarding rows in the standard manner and loading data into the production table.

If you choose to terminate the LOAD utility job and revert to the original table as it was before the LOAD utility began, you must terminate the utility ID by using the DB2 UET ABPMAINT utility program with the TERM_UTILITY option.

Usage considerations and restrictions

Consider the following usage issues for the IFDISCARDS option:

- This option requires DB2 version 10 or later.
- When an image copy is requested during the LOAD process, this option requires the following DB2 PTFs:
  - DB2 10: UI30114
  - DB2 11: UI30115
- When using the LOG NO option, specify the NOCOPYPEND option so that the LOAD utility does not set the table space in COPY-pending status. For more information, see the "Syntax and options of the LOAD control statement" topic in the *DB2 Utility Guide and Reference* documentation.
- For versioned objects, if you specify the REPLACE option, the current version number is reset to zero.
- Do not use this option if a DB2 utility is running against the same object. DB2 UET must stop the object briefly, and the utility might end with a "RESOURCE NOT AVAILABLE" error.
- This option does not support the following parallel processing:

- Parallel processing with a native DB2 LOAD job against the same partitions of a table space, including partitions implied by global options.
- Multiple jobs accessing the same partitions of a partitioned table space.

- If you specify this option and the job abends during the load, complete the following manual steps:
  1. Reset the status of the source objects to read-write (RW).
  2. Drop the shadow objects.
  3. Use the ABPMAINT utility to terminate the utility. For more information, see "Terminating a DB2 utility for which interception has occurred" on page 318.
- DB2 UET does not perform referential integrity checks during load prevalidation processing. If it detects referential integrity constraints, after a successful load, DB2 UET places the affected production objects in check-pending status.

The IFDISCARDS option cannot be used in conjunction with the following options and objects:

- DISCARDS *integer* with an *integer* value greater than 0

  If specified, then the job ends with return code 8. The utility job and its corresponding ID are terminated and cannot be restarted.
- SHRLEVEL CHANGE

  If specified, then the job ends, the LOAD utility ID is terminated, and an error message is issued.
- LOG YES
- BY PARTITION for spaces with nonpartitioned secondary indexes (NPSI)

  If BY PARTITION is specified, then the job ends and an error message is issued. You can load the entire table instead of partitions.
- LOB and XML objects
- VCAT-defined table spaces
- simple table spaces
- table spaces with rotated partitions
- RESUME NO (Use RESUME YES or REPLACE.)
- RESUME YES for table spaces with versioned rows

Related tasks:

"Terminating a DB2 utility for which interception has occurred" on page 318
If you need to terminate a DB2 utility for which DSNUTILB intercept processing is occurring or has occurred, use the ABPMAINT utility that is provided by DB2 UET.

Loading a table space in RO access mode (SHRLEVEL REFERENCE option)

When you specify the SHRLEVEL REFERENCE option, during the load, the table space is available in read-only mode.

When the LOAD utility job includes the SHRLEVEL REFERENCE option, processing is as follows:

1. DB2 UET creates a new set of shadow objects against which to perform the load.

   Shadow objects are duplicates that mirror the original database, table spaces, tables, and indexes. Shadow tables are created with same OBID as the original.

DB2 UET creates a unique name for the shadow objects by using the prefix that is specified in the option **Shadow database prefix** and adding a numeric suffix. In the shadow database, DB2 UET creates shadow copies of the table space, table, and all indexes on the table that is being loaded, and uses the value of the **Shadow schema** option as the schema name for the objects.

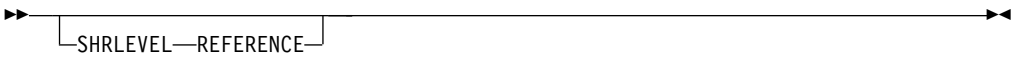
2. DB2 UET copies check constraints from production to shadow objects and determines the data set list of the objects that are being loaded.
3. DB2 UET places the production (original) objects in read-only access mode during processing.
4. If you specify RESUME YES, the contents of the existing production data sets are copied to shadow data sets (FlashCopy is used, if available). After copying the data DB2 UET runs a repair utility on the shadow objects to make the OBID, version, and level information of the copied data sets match the production objects.

   During this process, DB2 UET places the production object in read-only (RO) mode and briefly stops the shadow objects (allowing no access).
5. If an inline image copy was requested, the image copy is removed from the LOAD utility syntax.
6. DB2 UET loads the data into the shadow table instead of the production table.

   During the load, the *production* table is available for select processing.
7. After a successful LOAD utility job, processing is as follows:
   a. DB2 UET examines the shadow objects and repairs identifiers as necessary to match production objects.
   b. If an inline copy was requested and removed from the syntax, DB2 UET invokes the COPY utility to take a full image copy after the LOAD utility completes.
   c. A brief outage begins when DB2 UET stops the production spaces.
   d. To make the loaded data accessible in the new production object, DB2 UET swaps the underlying VSAM data sets for the production and shadow objects.
   e. DB2 UET updates real-time statistics (RTS) for the production table.
   f. If referential constraints on the old production table were detected, DB2 UET sets CHECK-pending (CHKP) status on the new production object, if appropriate.
   g. DB2 UET starts the new production object in the state that it was in before the load, and the outage ends.
   h. DB2 UET deletes renamed temporary data sets and drops the old production objects.

Syntax diagram

The following syntax diagram illustrates the correct syntax to use for adding the SHRLEVEL REFERENCE option to a DB2 LOAD utility statement that contains the REPLACE or RESUME NO option. (This syntax diagram is a fragment of the LOAD syntax diagram and does not show other native LOAD options that can be specified at this level.)



Usage considerations and restrictions

Consider the following usage issues for the SHRLEVEL REFERENCE option:

- This option requires DB2 version 10 or later.
- This option requires the DB2 COPY utility APAR PI43298.
- When an image copy is requested during the LOAD process, this option requires the following DB2 PTFs:
  - DB2 10: UI30114
  - DB2 11: UI30115
- Because DB2 UET requires random access to the image copy during the load process, the image copy cannot be on tape. (After the load, you can move the image copy back to tape.)
- When using the LOG NO option, specify the NOCOPYPEND option so that the LOAD utility does not set the table space in COPY-pending status. For more information, see the "Syntax and options of the LOAD control statement" topic in the *DB2 Utility Guide and Reference* documentation.
- When an inline image copy is requested, the image copy is taken against the shadow objects using its table space DBID and OBID. DB2 UET ensures that the image copy is updated so that it is valid for the production object.
- For versioned objects, if you specify the REPLACE option, the current version number is reset to zero.
- Do not use this option if a DB2 utility is running against the same object. DB2 UET must stop the object briefly, and the utility might end with a "RESOURCE NOT AVAILABLE" error.
- This option does not support the following parallel processing:
  - Parallel processing with a native DB2 LOAD job against the same partitions of a table space, including partitions implied by global options.
  - Multiple jobs accessing the same partitions of a partitioned table space.
- If you specify this option and the job abends during the load, complete the following manual steps:
  1. Reset the status of the source objects to read-write (RW).
  2. Drop the shadow objects.
  3. Use the ABPMAINT utility to terminate the utility. For more information, see “Terminating a DB2 utility for which interception has occurred” on page 318.
- DB2 UET does not perform referential integrity checks during load prevalidation processing. If it detects referential integrity constraints, after a successful load, DB2 UET places the affected production objects in check-pending status.

The SHRLEVEL REFERENCE option cannot be used in conjunction with the following options and objects:

-
- PART RESUME NO
- LOG YES
- BY PARTITION for spaces with nonpartitioned secondary indexes (NPSI)

  If BY PARTITION is specified, then the job ends and an error message is issued. You can load the entire table instead of partitions.
- LOB and XML objects
- VCAT-defined table spaces
- simple table spaces

- table spaces with rotated partitions
- RESUME NO (Use RESUME YES or REPLACE.)
- RESUME YES for table spaces with versioned rows

Using DFSMSdss COPY options

The following extended syntax options are available for use with the DFSMSdss COPY command when you specify LOAD RESUME or you specify LOAD REPLACE with the KEEPDICTIONARY option for a compressed table space:

FCTOPPRCP

DB2 UET gathers the value for this option from the ZPARM parameter FLASHCOPY PPRC. The FLASHCOPY PPRC parameter specifies the behavior for DFSMSdss FlashCopy® requests when the target disk storage volume is the primary device in a peer-to-peer remote copy relationship. FLASHCOPY_PPRC determines

- whether DFSMSdss preserves mirroring while processing a DB2 utilities request
- whether the target device pair is allowed to go to duplex pending state

For more information, see the IBM reference information for DB2 10 for z/OS online utilities.

Syntax diagram

```
>>-+---------------------------------------+----------------><
   '-FCTOPPRCP-----------------------------'
```

COPY_FASTREP

Specifies whether the use of DFSMSdss fast replication (FlashCopy®) is preferred, required, or not to be used. The value specified for this option does not affect concurrent copy or virtual concurrent copy processing. Valid values are as follows:

- PREFERRED: (default) Specifies that you want to use a fast replication method, if possible. If fast replication cannot be used, DFSMSdss completes the operation using traditional data movement methods.
- REQUIRED: Specifies that fast replication must be used. DFSMSdss stops processing the current data set if fast replication cannot be used. However, DFSMSdss continues processing the rest of the data sets using fast replication. When the DEBUG(FRMSG(MIN|SUM|DTL)) keyword is not specified, DFSMSdss still issues summarized information regarding why a fast replication method cannot be used as though DEBUG(FRMSG(SUMMARIZED)) had been specified. The DEBUG(FRMSG(MIN | SUM | DTL)) keyword determines the amount of information provided for why you cannot use a fast replication method.
- NONE: Specifies that fast replication should not be used. DFSMSdss does not attempt to use fast replication and completes the operation using traditional data movement methods.

For more information about FASTREPLICATION, see the explanation of COPY command keywords in the IBM *z/OS DFSMSdss Storage Administration* documentation.

Syntax diagram

```
    .---COPY_FASTREP----(--PREFerred--)---------.
>>--+---------------------------------------+-----------><
```

```
'-+-COPY_FASTREP-+--(--+-PREFerred--+--)----.
                       +-REQuired---+
                       '-NOne-------'
```

COPY_DEBUG

Specifies whether you want to use DEBUG as a diagnostic tool. Valid values are as follows:

- NONE: DEBUG is not used.
- FRMSG: DFSMSdss issues messages that explain why you cannot use fast replication or Preserve Mirror operation during COPY processing. Specify DEBUG(FRMSG) with one of the following sub-keywords:
  - FRMSG(MINIMAL): DFSMSdss issues a message with a minimal level of information.
  - FRMSG(SUMMARIZED): DFSMSdss issues an informational message with summary information.
  - FRMSG(DETAILED): DFSMSdss issues a message with detailed information When applicable, detailed information regarding ineligible volumes is provided in the message text.
- SMSMSG: DFSMSdss displays ACS WRITE statements to the job output.

For more information about DEBUG, see the explanation of COPY command keywords in the *IBM z/OS DFSMSdss Storage Administration* documentation.

Syntax diagram

```
>>-+-------------------------------------------------------------------+-><
   '-COPY_DEBUG--(--+--FRMSG----(--+-MINimal----+--)--+----------+-+--)-'
                    |              +-SUMmarized-+     '-,SMSMSG-'   |
                    |              +-DETAILED---+                    |
                    |              '-DTL--------'                    |
                    '--SMSMSG--+---------------------------------+-'
                               '-,FRMSG--(--+-MINimal------+--)--'
                                           +-SUMmarized---+
                                           +-DETAILED-----+
                                           '-DTL----------'
```

Determining whether the LOAD utility enhancements were implemented

To determine whether the DB2 UET options for the LOAD utility were processed, check the DB2 UET messages in the SYSPRINT data set for the LOAD utility. Use SDSF or an equivalent tool to view this information.

Look for the following messages in the SYSPRINT data set for the utility. These messages appear in the order shown. The message ABPU5331I presents the enhanced DSNUTILB SYSIN for the LOAD utility, including the DB2 UET options.

```
ABPU5330I date time Original DSNUTILB syntax follows:
ABPU5331I date time DSNUTILB SYSIN for the utility
ABPU5332I 194 16:52:18.42 End of original DSNUTILB syntax listing.
```

Tip: In addition to these messages, look for the messages that apply to DSNUTILB intercept processing in general. These messages are available in the SYSPRINT data set for the utility and in the SYSPRINT data set for the started task. For more information, see "Determining whether DSNUTILB intercept processing occurred" on page 309.

Related concepts:

"Determining whether DSNUTILB intercept processing occurred" on page 309
You can check whether DSNUTILB intercept processing occurred as you expected for DB2 utilities by checking the DB2 UET messages that are incorporated into the SYSPRINT data set for the utility job and the SYSPRINT data set for the DB2 UET started task. Use SDSF or an equivalent tool to view this information.

Related tasks:

"Implementing the LOAD utility enhancements" on page 268
Perform the following steps to implement any of the additional options that DB2 UET supplies for the LOAD utility.

Chapter 9. Automatically creating mapping tables (REORG TABLESPACE utility enhancements)

DB2 UET can automatically size and create the mapping table and mapping-table index that are required for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option. When the reorganization completes, these objects are automatically dropped to preserve space.

This enhancement is implemented through the use of the DSNUTILB intercept. You must create a DSNUTILB intercept policy that specifies the DB2 subsystem or subsystems on which to enable the automatic creation of mapping tables.

DB2 UET can automatically size and create a mapping table and mapping-table index whenever DSNUTILB invokes the REORG TABLESPACE utility with the SHRLEVEL CHANGE option. DB2 UET inserts the DDL for creating the mapping table and mapping-table index. The DDL directs DB2 to implicitly create a unique database and table space for each concurrent utility job. DB2 UET also inserts the DDL for dropping these objects at the appropriate locations within the input data set that DSNUTILB receives.

With this enhancement, you do not need to manually specify the MAPPINGTABLE option on the SHRLEVEL CHANGE parameter because DB2 UET automatically adds it, according to the DB2 REORG TABLESPACE syntax rules. (For a description of these rules, see the *DB2 for z/OS Utility Guide and Reference*.) The MAPPINGTABLE option specifies the name of the generated mapping table. DB2 UET generates this name by using the pattern *user_id*.*utility_id*, where *user_id* is the user ID under which the REORG TABLESPACE utility is running and *utility_id* is the UTILID for that utility execution. To generate the name of the mapping-table index, DB2 UET uses the same pattern with the literal value "IX" at the end, that is, *user_id*.*utility_id*.IX.

If a MAPPINGTABLE specification already exists in the input data set for the utility, DB2 UET will either retain or replace the existing MAPPINGTABLE specification depending on how you set the OVERRIDE_MAPPING_TABLE started task initialization option. If you set the OVERRIDE_MAPPING_TABLE option to YES, DB2 UET will attempt to override the existing MAPPINGTABLE specification prior to invoking DSNUTILB to create a mapping table. If you set the option to NO, DB2 UET will not attempt to override the existing MAPPINGTABLE specification.

Note: DB2 Version 11 and later: The MAPPINGTABLE and MAPPINGDATABASE parameters are mutually exclusive. After a subsystem has been migrated to DB2 V11 or later, if MAPPINGDATABASE is specified, then DB2 UET does not add MAPPINGTABLE to the syntax.

Topics:

Related concepts:

"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

Implementing the REORG TABLESPACE utility enhancements

Perform these steps to have DB2 UET automatically create and drop mapping tables and mapping-table indexes for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option.

Before you begin

Ensure that the started task initialization option OVERRIDE_MAPPING_TABLE is set appropriately. To replace any existing MAPPINGTABLE specification that you manually defined for the utility with the MAPPINGTABLE specifications that DB2 UET can generate, set the OVERRIDE_MAPPING_TABLE option to YES.

Ensure that you have current statistics for the object to be reorganized so that the mapping table is sized appropriately.

About this task

For more information, see "Started task initialization options" on page 98.

Procedure

1. In your DSNUTILB intercept policy, specify the DB2 subsystem or subsystems on which you want DB2 UET to automatically generate mapping tables and mapping-table indexes for the REORG TABLESPACE utility. You specify a subsystem in the <POLICY> section by using the <DB2SYSTEM> element. For more information, see "Creating a DSNUTILB intercept policy" on page 160.
2. Ensure that the DSNUTILB intercept status is Enabled. You can display the intercept status by issuing the DISPLAY INTERCEPT operator command from the z/OS console. If the intercept is disabled, you will need to activate it by issuing the ACTIVATE INTERCEPT console command. For more information about these console commands, see "Console commands for the started task" on page 447.

Results

When DSNUTILB invokes the REORG TABLESPACE utility with the SHRLEVEL CHANGE option, DB2 UET will dynamically generate the mapping table and mapping-table index. After the REORG TABLESPACE utility completes, these objects will be automatically dropped.

What to do next

You can check the SYSPRINT data set for the REORG TABLESPACE utility to determine if the mapping-table objects were created. See "Determining whether the REORG TABLESPACE utility enhancements were implemented" on page 293.

Related concepts:

"Determining whether the REORG TABLESPACE utility enhancements were implemented"
To determine whether DB2 UET was able to generate the mapping table and mapping table index for the REORG TABLESPACE utility, check the DB2 messages in the SYSPRINT data set for the utility. The DB2 UET SYSPRINT displays message ABPU5407I to indicate that the mapping table was created successfully.

"Defining and using a DSNUTILB intercept policy" on page 129
If you want to use the DSNUTILB intercept component to implement the DB2 UET enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads automatically for certain DB2 utility operations, you must define a DSNUTILB intercept policy.

Determining whether the REORG TABLESPACE utility enhancements were implemented

To determine whether DB2 UET was able to generate the mapping table and mapping table index for the REORG TABLESPACE utility, check the DB2 messages in the SYSPRINT data set for the utility. The DB2 UET SYSPRINT displays message ABPU5407I to indicate that the mapping table was created successfully.

For example:

```
ABPU5407I 124 07:27:18.04 SQL CREATE successful for
mapping table CSTROS."A142606.REORG"
```

Tip: In addition to these messages, you can look for the messages that apply to DSNUTILB intercept processing in general. These messages are available in the SYSPRINT data set for the utility and in the SYSPRINT data set for the started task. For more information, see "Determining whether DSNUTILB intercept processing occurred" on page 309.

Related concepts:

"Determining whether DSNUTILB intercept processing occurred" on page 309
You can check whether DSNUTILB intercept processing occurred as you expected for DB2 utilities by checking the DB2 UET messages that are incorporated into the SYSPRINT data set for the utility job and the SYSPRINT data set for the DB2 UET started task. Use SDSF or an equivalent tool to view this information.

Related tasks:

"Implementing the REORG TABLESPACE utility enhancements" on page 292
Perform these steps to have DB2 UET automatically create and drop mapping tables and mapping-table indexes for the REORG TABLESPACE utility when it is invoked by DSNUTILB with the SHRLEVEL CHANGE option.

Related reference:

"DB2 Utilities Enhancement Tool messages" on page 375
Look up DB2 UET messages to obtain information about them, including message explanations and suggested responses.

Chapter 10. Substituting HPU for the DB2 UNLOAD utility

DB2 UET can substitute the IBM DB2 High Performance Unload for z/OS (HPU) product for the DB2 UNLOAD utility.

After you implement the enhancement, the UNLOAD utility is changed to an HPU unload with no alterations to the syntax other than those that you specify in element <MONITOR>. By using a policy that indicates that DB2 UET is to replace DB2 UNLOAD for HPU, DB2 UET treats the syntax as a native DB2 UNLOAD utility. Normal syntax validation occurs through DSNUTILB. DB2 UET invokes HPU, and processing is controlled by HPU options.

Considerations

- Any utility syntax that is used must comply with both native DB2 UNLOAD utility and High Performance Unload (HPU) utility syntax requirements.
- DB2 UET does not convert DB2 UNLOAD utility syntax to HPU syntax.
- For more information about DB2 High Performance Unload for z/OS (HPU), see the following Web page:

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2tools.inz42.doc.ug%2Finzhome.htm

Implementing the UNLOAD utility enhancement

To use the enhancement, complete the following steps:

1. Install IBM DB2 High Performance Unload for z/OS (HPU) on the DB2 subsystem on which unload utilities are to be substituted for HPU.
2. Add the HPU SINZLINK library to either the UNLOAD job STEPLIB or the DB2 UET started task (STC) STEPLIB.

   See Figure 33 on page 296 and Figure 34 on page 296.

   Note: If you add HPU SINZLINK to the DB2 UET STC STEPLIB, you do not need to modify UNLOAD job JCL to substitute HPU for the UNLOAD utility.
3. Include the HPU load library in your system APF-authorized list.
4. Provide a policy that includes attribute <REPLACE> on element <UTILITY>, as shown in Figure 35 on page 296.

Examples

Note: Any utility syntax that is used must comply with both native DB2 UNLOAD utility and HPU utility syntax requirements.

```
//*****************************************************************
//*    UNLOAD
//*****************************************************************
//*
//UNLOAD   EXEC PGM=DSNUTILB,
//             PARM='ssid,utilid'
//*
//* INVOKE DB2 UET INTERCEPT
//*
//STEPLIB  DD  DISP=SHR,DSN=DSN.SDSNLOAD
//*
//UNLDD    DD DSN=unload.dsn,
//*           UNIT=SYSDA,
//*           SPACE=(CYL,(2,2)),
//            DISP=(NEW,CATLG,CATLG)
//*
//SYSIN    DD  *
   UNLOAD TABLESPACE DBNAME.TSNAME
   UNLDDN UNLDD
/*
//SYSPRINT DD  SYSOUT=*
```

Figure 33. Example: UNLOAD utility JCL

```
//ABPSTC   EXEC PGM=ABPSINIT,TIME=1440,REGION=0M
//*
//STEPLIB  DD  DISP=SHR,DSN=ABP.SAPLOAD
//         DD  DISP=SHR,DSN=DSN.SDSNLOAD
//         DD  DISP=SHR,DSN=HPU.SINZLINK
.
```

Figure 34. Example: Started task (STC) STEPLIB

```
<PRACTICE NAME="STANDARDS_1">
        <UTILITY NAME="UNLOAD" REPLACE="HPU">
                .
                .
        </UTILITY>
</PRACTICE>

<POLICY>
       <DB2SYSTEM SSID="LAA" ACTION="MONITOR_UTILITY">
              <USE_PRACTICE NAME="STANDARDS_1"/>
              <INCLUDE>
                     <RULE DATABASE="PROD"/>
                     <RULE TABLESPACE="BIG%"/>
              </INCLUDE>
       </DB2SYSTEM>
</POLICY>
```

Figure 35. Example policy

Related reference:

"<UTILITY>" on page 156
The element <UTILITY>, when used in conjunction with the attribute NAME, identifies the DB2 utility that DB2 UET is to evaluate when DSNUTILB is invoked to verify whether the Practice rule should be enforced.

Chapter 11. Administering the product

If you are the DB2 UET administrator, you will need to occasionally perform some product administration tasks to manage and maintain the product.

You can perform the following tasks from the DB2 UET ISPF interface if you have the authority that is needed to access the Administrator Functions menu and panels:

- Viewing status information for the DB2 UET system that you are currently using, including the settings for the started task initialization options. For example, you might want to check whether the logging of messages is enabled or the name of the DB2 plan that is being used for the product configuration.
- Overriding some started task initialization options temporarily. For example, you might want to route WTO messages to another z/OS console or enable the recording of trace information.

Tip: You can control which users can access the administrator panels and override started task initialization options by using a security exit.

Occasionally, you might need to perform the following tasks from SDSF or an equivalent tool:

- Stopping the started task
- Manually performing cleanup of the DB2 tables for audit, logging, and DSNUTILB-intercept worklist error information if the automatic cleanup of these tables is disabled
- Querying the audit and logging tables for the information that you need for audit or diagnostic use

Also, if you encounter a problem, you might need to produce a dump file and send diagnostic information to Support. For more information, see the Chapter 13, “Reference,” on page 441 topics.

Topics:

Viewing system status

You can view current information about the DB2 UET system that you are using, including the ABPID (product configuration ID), the started task name, and the values that are set for the started task initialization options.

About this task

You set the started task initialization options in the SABPSAMP member *abpid*OPTS during product customization. If you omitted an option from the initialization

options member or did not provide a value for an option, the Display System Status panel will display the default value for that option, if one is available, or a blank. If you specified temporary override values for some initialization options from the Control System panel, the Display System Status panel will display your temporary override values instead of those from the initialization options member.

Procedure

1. From the Main Menu, choose option **3** (Administrator functions). The Administrator Functions panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   Administrator Functions              11:34:42
Option ===>

  1  Display Utilities Enhancement Tool system status  ABPID . . . : ABP1
  2  Control Utilities Enhancement Tool system         DB2 system. : DBP1
                                                       User ID . . : PDABCD
```

Figure 36. Administrator Functions panel

2. Choose option **1** (Display DB2 UET system status). The Display System Status panel is displayed.

   Tip: To ensure that you are viewing the latest settings, you can type `REFRESH` at the command line and press Enter. Also, you can display Help information for any field on this panel by moving your cursor to the field and pressing F1.
3. Review the information on the panel. For more information about the fields, see "Display System Status panel fields."
4. When you are finished, press F3.

Display System Status panel fields

This reference section provides more information about the Display System Status panel fields.

ABPID
: Displays the identifier for the DB2 UET configuration to which you are connected. You specified this ID in the started task initialization options member by using the ABPID option. Also, you selected this ID on the Set ABPID panel.

Started task name
: Displays the name of the DB2 UET started task that is running.

Started task ASID
: Displays the address space identifier (ASID), in hexadecimal format, for the address space in which the started task is running.

Start timestamp
: Displays the system timestamp for the point in time at which the started task last started. The timestamp is in the format YYYY-MM-DD-HH:MM:SS, where *YYYY* is the year, *MM* is the month, *DD* is the day, *HH* is hours, *MM* is minutes, and *SS* is seconds.

ABPBMAIN cancel syntax member
: Displays the name of SABPSAMP member that contains the optional cancel-control parameters for any thread cancelations that you perform by using the DSNUTILB intercept. During customization, DB2 UET created and customized the member *abpid*BCAN (where *abpid* is the **ABPID** value) in the SABPSAMP library for your use. You specify the member name in

the started task initialization options member by using by the ABPBMAIN_CANCEL_MEMBER option.

ABPBMAIN global syntax member
Displays the name of the SABPSAMP member that contains the optional global parameters for any thread cancelations that you perform by using the DSNUTILB intercept. During customization, DB2 UET created and customized the member *abpid*BGLB (where *abpid* is the **ABPID** value) in the SABPSAMP library for your use. You specify the member name in the started task initialization options member by using by the ABPBMAIN_GLOBAL_MEMBER option.

ABPOPTS data set name
Displays the name of the partitioned data set (PDS) that contains the started task initialization options member. The member name is displayed in the **ABPOPTS member name** field.

ABPOPTS member name
Displays the name of the PDS member that contains your started task initialization options. This member is in the PDS that is specified by the **ABPOPTS data set name** field. During customization, DB2 UET created and customized the member *abpid*OPTS (where *abpid* is the **ABPID** value) in the SABPSAMP library for your use.

ABPPLCY data set name
If you defined the DSNUTILB intercept policy in a sequential file, this field displays the name of that file. If you defined the policy in a PDS member, this field displays the name of the PDS that contains that member, and the **ABPPLCY member name** field displays the member name.

ABPPLCY member name
Displays the name of the PDS member that contains your DSNUTILB intercept policy. This member is in the PDS that is specified in the **ABPPLCY data set name** field. During customization, DB2 UET created and customized the sample policy member *abpid*PLCY (where *abpid* is the **ABPID** value) in the SABPSAMP library for your use. If you defined the policy in a sequential file instead, this field is blank and the **ABPPLCY data set name** field displays the file name.

Audit status
Indicates whether DB2 UET records audit information for thread-cancelation activities in its DB2 audit table. Valid values are: ACTIVE (records audit information) and INACTIVE (does not record audit information). You set this value in the started task initialization options member by using the AUDIT_ACTIVE option. You can specify a temporary override value on the Control System panel.

Audit rows maximum age
Displays the maximum number of days that the audit table (ABPAUDIT) rows are retained. This number of days is counted from the time when the rows are inserted into the table. You specify this value in the started task initialization options member by using the AUDIT_MAX_AGE option. If the value is greater than zero, when a row reaches the specified age limit, it is automatically deleted from the table. If the value is zero (the default value), rows are *not* automatically deleted from the table; you must manually delete them periodically to prevent the table from becoming too large. To do so, you can use the sample SQL in the SABPSAMP member ABPAUDD.

Audit table name
Displays the name of the DB2 table that contains the DB2 UET audit information for thread cancelations. This value is in the format *creator_id*.*table_name*. The default table name is ABPAUDIT. The table resides on the DB2 subsystem that is specified in the **DB2 system for log and audit** field.

Cancel escalation active
Displays whether users are allowed to perform an escalated cancelation of a thread. For an escalated cancelation, DB2 UET issues the z/OS cancel command to terminate the job, z/OS user, or started task that is associated with the thread. The value YES indicates that escalated cancelations are allowed, and the value NO indicates that they are not allowed (only the DB2 -CANCEL THREAD command is used). You specify this value in the started task initialization options member by using the CANCEL_ESCALATION_ACTIVE option.

DB2 connect to all subsystems
Indicates whether DB2 UET will attempt to connect to all active DB2 subsystems on the z/OS system or only to the DB2 subsystem that is specified by the DB2_SSID option in the started task initialization options member (the subsystem that contains audit and logging information). If YES (the default value) is specified, DB2 UET will attempt to connect to all active DB2 subsystems. If NO is specified, DB2 UET will attempt to connect only to the DB2 subsystem that contains audit and logging information. You specify this value in the started task initialization options member by using the DB2_CONNECT_TO_ALL_SUBSYSTEMS option.

DB2 connection idle timeout
Displays the maximum amount of time (in seconds) that the DB2 connection for a DB2 UET task can have no activity. When this time limit is reached, the connection to DB2 closes. If the value is zero, this timeout limit is disabled and will not cause an inactive connection to close. You specify this value in the started task initialization options member by using the DB2_CONNECTION_IDLE_TIMEOUT option.

DB2 plan name
Displays the name of the DB2 plan that DB2 UET is using to access its DB2 tables. You set this name in the started task initialization options member by using the DB2_PLAN_NAME option.

DB2 system for log and audit
Displays the subsystem identifier (SSID) of the DB2 subsystem that contains the DB2 UET tables for log and audit information. You set this SSID in the started task initialization options member by using the DB2_SSID option.

DB2 task count
Displays the maximum number of z/OS tasks that DB2 UET can start for connection to a single DB2 subsystem. You set this value in the started task initialization options member by using the DB2_TASK_COUNT option.

DB2 task idle timeout
Displays the maximum amount of time (in seconds) that a subtask for a DB2 UET connection to DB2 can remain inactive after the connection closes (after the **DB2 connection idle timeout** limit is met). When this task-idle time limit is reached, the subtask ends. If the value is zero, this timeout

limit is disabled and will not cause an inactive subtask to end. You set this value in the started task initialization options member by using the DB2_TASK_IDLE_TIMEOUT option.

Dynamic SYSOUT class
Displays the JES SYSOUT class for the SYSOUT data sets that DB2 UET dynamically allocates during DSNUTILB interception for the SYSPRINT output for a utility job. The default value is an asterisk (*), which indicates that DB2 UET should use the default SYSOUT class that is specified for the job, started task, or TSO session under which DSNUTILB is running. You set this value in the started task initialization options member by using the DYNAMIC_SYSOUT_CLASS option. If you have an output management product that captures and deletes SYSOUT data sets automatically, you should have set this value to a SYSOUT class that your output management product will *not* delete. Otherwise, DSNUTILB interception errors might occur. Regardless of how you set this option, note that the following dynamically allocated SYSOUT data sets will always use the default asterisk (*) class: the SYSOUT data sets that are dynamically allocated for the output from the batch interface (ABPBMAIN) and the ABPSORT data sets (which are used in sort processing for the DSNUTILB intercept).

Note: DYNAMIC_SYSOUT_CLASS="*class*" should be customized for JES3. Using the default value (*) is not recommended in a JES3 environment. Instead, in a JES3 environment, make sure that you set this option to a SYSOUT class that is defined with the HOLD=TSO parameter. If this option is set to a SYSOUT class that is not defined with the HOLD=TSO parameter, the DSNUTILB intercept will not be able to recombine SYSOUT files that are produced by the DB2 UET and the DSNUTILB utility. In this case, the SYSOUT will show up in the JES3 spool as multiple files. Some of the files will be named SYSPRINT and others will have a system-generated file name such as SYS*nnnn*

Logging status
Indicates whether DB2 UET logs messages about product operations and performance in its DB2 log table. Valid values are: ACTIVE (logs messages) and INACTIVE (does not log messages). You set this value in the started task initialization options member by using the LOGGING_ACTIVE option. You can specify a temporary override value on the Control System panel.

Log rows maximum age
Displays the maximum number of days that rows in the logging table (ABPLOG) are retained. This number of days is counted from the time when the rows are inserted into the table. You specify this value in the started task initialization options member by using the LOGGING_MAX_AGE option. If the value is greater than zero, rows are automatically deleted from the table when they reach this age limit. If the value is zero (the default value), rows are *not* automatically deleted from the table; you must manually delete them periodically to prevent the table from becoming too large. To do so, you can use the SQL in the SABPSAMP member ABPLOGD.

Logging table name
Displays the name of the DB2 table that contains the DB2 UET logged messages. This value is in the format *creator_id.table_name*. The default table name is ABPLOG. This table resides on the subsystem that is specified in the **DB2 system for log and audit** field.

Work bufferpool name
Displays the name of the buffer pool that is used for work objects that are created for the extended functionality of the CHECK DATA utility. You specify this value in the started task initialization options member by using the WORK_DATABASE_BUFFERPOOL option.

Work database name
Displays the name of the database in which you store work objects that are created for the extended functionality of the CHECK DATA utility. You specify this value in the started task initialization options member by using the WORK_DATABASE_NAME option.

Work stogroup name
Displays the name of the DB2 STOGROUP that is used for work objects that are created for the extended functionality of the CHECK DATA utility. You specify this value in the started task initialization options member by using the WORK_DATABASE_STOGROUP option.

Override mapping table
Indicates whether any existing MAPPINGTABLE specifications that you manually defined for the DB2 REORG TABLESPACE utility are replaced by the MAPPINGTABLE specifications that DB2 UET can automatically generate for the utility when the SHRLEVEL CHANGE option is specified. If the value is YES, the replacement occurs. If the value is NO, the replacement does not occur. You specify this value in the started task initialization options member by using the OVERRIDE_MAPPING_TABLE option.

Post-cancel exit name
Displays the name of the user exit that you optionally created to perform some processing that you determined is necessary after thread cancelations. You specify the exit name in the started task initialization options member by using the POST_CANCEL_EXIT option. If you are not using a post-cancel exit, the value NONE should be displayed.

Pre-cancel exit name
Displays the name of the user exit that you optionally created to perform some processing that you determined is necessary prior to thread cancelations. You specify the exit name in the started task initialization options member by using the PRE_CANCEL_EXIT option. If you are not using a pre-cancel exit, the value NONE should be displayed.

Security exit name
Displays the name of the security user exit that you optionally created to control user access to thread-management functions such as blocking and canceling threads and to specific ISPF panels. You specify the exit name in the started task initialization options member by using the SECURITY_EXIT option. If you are not using a security exit, the value NONE should be displayed. [

SVC number
Displays the number that identifies the DB2 UET supervisor call (SVC). The SVC is dynamically installed when the started task starts and dynamically removed when the started task stops. It is required for the product interfaces to communicate with the started task. You specify this value in the started task initialization options member by using the SVC_NUMBER option.

Tracing status
Indicates whether DB2 UET records trace information. A trace is a record

of internal processing that is primarily used by Support for diagnosing a problem. Valid values are: ACTIVE (records trace information) and INACTIVE (does not record trace information). You specify this value in the started task initialization options member by using the TRACE_ACTIVE option. You can specify a temporary override value on the Control System panel.

Trace table size (MB)
Displays the size (in MB) of the table in which DB2 UET stores trace information. You specify this value in the started task initialization options member by using the TRACE_SIZE option. Valid values are 1 - 2147483647. The default value is 1. A value of 0 will result in no trace table allocation.

Wildcard - single character
Displays the wildcard character that represents a single character: an underscore (_). You can use this wildcard character in any field that supports wildcards.

Wildcard - 0 or more characters
Displays the wildcard character that represents zero or more characters: a percent sign (%). You can use this wildcard character in any field that supports wildcards.

Workfile DATACLAS name
Displays an SMS data class for the temporary DASD data sets that are allocated by DB2 UET, or the value NONE. You specify this value in the started task initialization options member by using the WORKFILE_DATACLAS option. You can specify a temporary override value on the Control System panel.

Workfile MGMTCLAS name
Displays an SMS management class for the temporary DASD data sets that are allocated by DB2 UET, or the value NONE. You specify this value in the started task initialization options member by using the WORKFILE_MGMTCLAS option. You can specify a temporary override value on the Control System panel.

Workfile STORCLAS name
Displays an SMS storage class for the temporary DASD data sets that are allocated by DB2 UET, or the value NONE. You specify this value in the started task initialization options member by using the WORKFILE_STORCLAS option. You can specify a temporary override value on the Control System panel.

Workfile UNIT name
Displays the unit name for the location where the temporary DASD data sets that are allocated by DB2 UET are stored. This value can be any valid unit name for a storage device; the value VIO if VIO (virtual input/output) storage groups are supported on your system and you want the temporary data sets to reside entirely in paging storage to improve performance; or the value NONE. You specify this value in the started task initialization options member by using the WORKFILE_UNIT option. You can specify a temporary override value on the Control System panel.

Worklist error rows max age
Displays the maximum number of days that rows with diagnostic data are retained in worklist-error tables. A worklist contains enhanced SYSIN data for a DB2 utility and can be used for restart purposes. Worklist data is written to worklist-error tables for diagnostic use by Customer Support in certain situations. If zero is displayed, rows are *not* deleted from the

worklist-error tables based on this parameter; you might need to manually delete old rows from worklist-error tables periodically to prevent the tables from becoming too large. You specify this value in the started task initialization options member by using the WORKLIST_ERROR_MAX_AGE option.

WTO ROUTCDE
Displays the routing code for write-to-operator (WTO) messages regarding DB2 UET operations. A routing code is an integer from 1 through 28. Routing codes identify the z/OS console to which WTO messages are sent. You specify this value in the started task initialization options member by using in the WTO_ROUTCDE option. You can specify a temporary override value on the Control System panel.

Overriding started task initialization options

From the DB2 UET ISPF interface, you can temporarily override the values for some started task initialization options if you have the authority to do so

About this task

You can change the initialization options that control whether logging, audit, and trace information is recorded; options that affect how DB2 UET temporary data sets are allocated; and the option that specifies a routing code for write-to-operator (WTO) messages. Any changes that you make to these options will remain in effect until the DB2 UET started task is restarted. Your changes are *not* saved to the started task initialization options member (*abpid*OPTS). If you want your changes to persist, you must edit the started task initialization options member.

Procedure

1. From the Main Menu, choose option **3** (Administrator functions). The Administrator Functions menu panel is displayed.
2. Choose option **2** (Control system). The Control System panel is displayed, as follows:

```
DB2 Utilities Enhancement Tool   Control System                    Top of data
Command ===>

Commands: REFRESH  SAVE
                                                                   More:     +
ABPID. . . . . . . . . . . . . : ABP1

Started task name. . . . . . . : ABP1
Started task ASID. . . . . . . : 03EE
Start timestamp. . . . . . . . : 2015-12-02 10:28:04

ABPBMAIN cancel syntax member. : ABP1BCAN
ABPBMAIN global syntax member. : ABP1BGLB
ABPOPTS data set name  . . . . : ABP.MYLIBS.SABPSAMP
ABPOPTS member name. . . . . . : ABP1OPTS
ABPPLCY data set name. . . . . : ABP.MYLIBS.SABPSAMP
ABPPLCY member name. . . . . . : ABP1PLCY
Audit status . . . . . . . . . . ACTIVE        (Active or Inactive)
Audit rows maximum age . . . . : 0
Audit table name . . . . . . . : ABP.ABPAUDIT
Cancel escalation active . . . : NO
DB2 connect to all subsystems. : YES
DB2 connection idle timeout. . : 300
DB2 plan name. . . . . . . . . : PGMAPLAN
DB2 system for log and audit . : DBP1
DB2 task count . . . . . . . . : 2
DB2 task idle timeout. . . . . : 900
Dynamic SYSOUT class . . . . . : *
Logging status . . . . . . . . . ACTIVE        (Active or Inactive)
Log rows maximum age . . . . . : 0
Logging table name . . . . . . : ABP.ABPLOG
Work bufferpool name . . . . . : BP0
Work database name . . .  .  . : ABPGG01T
Work stogroup name . . . . . . : SYSDEFLT
Override mapping table . . . . : NO
Post-cancel exit name. . . . . : NONE
Pre-cancel exit name . . . . . : NONE
Security exit name . . . . . . : NONE
SVC number . . . . . . . . . . : 250
Tracing status . . . . . . . . . ACTIVE        (Active or Inactive)
Trace table size (MB). . . . . : 1
Wildcard - single character. . : _
Wildcard - 0 or more characters: %
Workfile DATACLAS name . . . . . NONE          (SMS DATACLAS or NONE)
Workfile MGMTCLAS name . . . . . NONE          (SMS MGMTCLAS or NONE)
Workfile STORCLAS name . . . . . NONE          (SMS STORCLAS or NONE)
Workfile UNIT name . . . . . . . VIO           (UNIT name or NONE)
Worklist error rows max age. . : 0
WTO ROUTCDE. . . . . . . . . . . 11            (0-15)
```

Figure 37. Control System panel

If you cannot access this panel, you might not have authority to do so. A security exit might be defined for your system that prevents you from accessing the administrative panels.

Tip: You can display Help information for each field on this panel by moving your cursor to a field and pressing F1.

3. In the **Logging status** field, type one of the following values to temporarily change whether messages from product operations are logged:
   - `Active` (or `A`) to enable logging
   - `Inactive` (or `I`) to disable logging
4. In the **Audit status** field, type one of the following values to temporarily change whether information for auditing thread cancelation activities is recorded:
   - `Active` (or `A`) to record audit information

- `Inactive` (or `I`) to not record audit information

5. In the **Trace status** field, type one of the following values to temporarily change whether trace information for diagnostic use is recorded:
   - `Active` (or `A`) to record trace information
   - `Inactive` (or `I`) to not record trace information
6. If you want to change how the DB2 UET temporary data sets are allocated, specify a value in any of the following fields:

   Workfile DATACLAS name
   Specify a valid SMS data class for the temporary DASD data sets that are allocated by DB2 UET, or specify `NONE`.

   Workfile MGMTCLAS name
   Specify a valid SMS management class for the temporary DASD data sets that are allocated by DB2 UET, or specify `NONE`.

   Workfile STORCLAS name
   Specify a valid SMS storage class for the temporary DASD data sets that are allocated by DB2 UET, or specify `NONE`.

   Workfile UNIT name
   Specify a unit name for the temporary DASD data sets that are allocated by DB2 UET. You can specify any valid unit name for a storage device (an address, a generic name, or a group name of a DASD device); the value `VIO` if VIO (virtual input/output) storage groups are supported on your system and you want the temporary data sets to reside entirely in paging storage to improve performance; or the value `NONE`.
7. In the **WTO ROUTCDE** field, type a routing code for WTO messages if you want to change this code. The routing code indicates the z/OS console to which WTO messages on DB2 UET activities will be sent. This value can be an integer from 1 to 28.
8. At the command line, type `SAVE` and press Enter to save your changes. If you are not authorized to make these changes, an error message is displayed.
9. Press F3.

Stopping the started task

Occasionally, you might to need to stop the DB2 UET started task to perform a task such as applying product maintenance.

Before you begin

Before stopping the started task, make sure that no DB2 utilities are running that use the started task. Stopping the started task when a DB2 utility is running will cause the utility job to end with return code 8. To list information on currently active sessions, issue the **`DISPLAY SESSIONS`** command as described in "Console commands for the started task" on page 447.

About this task

To stop the started task, issue one the following commands:

From the z/OS console:

```
P abpstc
F abpstc, stop
```

From SDSF:

```
/P abpstc
/F abpstc, stop
```

Where *abpstc* is the member name of the DB2 UET PROC in the PROCLIB.

During customization, the Tools Customizer generated the started task name based on the value that you specified in the **Configuration ID** field of the Install Customization - Variables dialog and inserted this name into the started task PROC. If you edited the started task name in the PROC, you must specify the new name in this command.

Related reference:

"Console commands for the started task" on page 447
DB2 UET supports several z/OS console commands for the started task. You issue these commands by using the Modify operator command.

Maintaining the product tables

If you set the started task initialization options AUDIT_MAX_AGE, LOGGING_MAX_AGE, and WORKLIST_ERROR_MAX_AGE to a non-zero value, DB2 UET will automatically delete rows that exceed the specified age limit from the audit, logging, and DSNUTILB-intercept worklist-error tables. However, if you set these options to zero, you might need to manually delete obsolete rows from these tables periodically to prevent the tables from becoming too large.

About this task

To do so, you can use the sample SQL that is provided in the following members of the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization:

- ABPAUDD to delete rows from the audit table.
- ABPLOGD to delete rows from the logging table.
- ABPWKED to delete rows from the DSNUTILB-intercept worklist-error tables. (Worklist data is moved to the worklist-error tables for diagnostic use by Customer Support only in certain situations.)

In each of these members, you specify a number of days prior to the current date. DB2 UET will then delete any rows that were inserted into the table *before* that number of days ago. For example, if you specify 60 days, all rows that were inserted 61 days ago and earlier will be deleted.

You do not need to delete information from the thread-blocker or worklist tables. DB2 UET automatically deletes rows for a thread-blocking operation from the thread-blocker tables when the ALLOW_THREADS action completes. DB2 UET automatically deletes rows for a particular utility ID from the worklist tables when intercept processing for that utility ID completes.

Also, you do not need to delete data from any mapping tables or mapping-table indexes that DB2 UET dynamically creates for the REORG TABLESPACE utility when the SHRLEVEL CHANGE option is specified. These objects are automatically dropped when REORG TABLESPACE processing completes.

Obtaining information from the audit and logging tables

You can create SQL queries to obtain information from the audit and logging tables for audit or troubleshooting purposes. For example, you could create a query to identify the users who were canceling threads during a certain period or to gather messages related to a specific operation for diagnostic use.

To create a query that obtains the information you need, you will need to understand the table formats.

For information about the table formats, see "Audit table format" and "Logging table format."

Audit table format

This topic describes the columns in the audit table (ABPAUDIT). You must become familiar with these columns to create an SQL query that gathers the information that you need from this table.

The audit table includes the following columns:

ABPID
The identifier for the DB2 UET configuration that was running

DATA 1-4
Columns that contain a keyword followed by data

ENVIRONMENT
The interface that was running. Valid values are:

- **I** for the ISPF interface
- **B** for the batch interface
- **M** for the ABPMAINT utility
- **U** for the DSNUTILB intercept

EVENT
One of the following event types:

- CANCEL THREAD
- CANCEL THREAD (ESCALATED)
- SIGNOFF
- SIGNON

INSERTED
The timestamp that indicates when the row was inserted into the table

SESSION
The interface session number

USERID
The user ID of the user who was performing the activity

Logging table format

This topic describes the columns in the logging table (ABPLOG). You must become familiar with these columns to create an SQL query that gathers the information that you need from this table.

The logging table includes the following columns:

ABPID
The identifier for the DB2 UET configuration that was running

INSERTED
The timestamp that indicates when the row was inserted into the table

MSGID
The message identifier (message number)

MSGTEXT
The message text that follows the message number

Managing DSNUTILB interception

You can manage DSNUTILB interception by performing some routine and occasional tasks.

On a routine basis, check the messages from utility processing to determine whether DSNUTILB interception occurred and whether the DB2 UET enhancements that you configured were correctly implemented. Occasionally, you might do other tasks, for example, check or change the activation status of the intercept, view the current intercept policy, diagnose interception problems, terminate a utility for which interception has occurred and clean up the associated worklist data, or restart a utility from the appropriate point when a normal DB2 restart fails.

Related concepts:

"Preparing to intercept the DSNUTILB program and DB2 utilities" on page 122
The DB2 UET DSNUTILB intercept is a front end to the DSNUTILB program and the DB2 utilities. You must use the intercept to implement the DB2 UET enhancements specifically for the LOAD utility and the REORG TABLESPACE utility. You can also use the intercept to block and cancel threads on DB2 objects that are needed for these and other DB2 utilities.

"Defining and using a DSNUTILB intercept policy" on page 129
If you want to use the DSNUTILB intercept component to implement the DB2 UET enhancements for the LOAD or REORG TABLESPACE utility or to block and cancel threads automatically for certain DB2 utility operations, you must define a DSNUTILB intercept policy.

Determining whether DSNUTILB intercept processing occurred

You can check whether DSNUTILB intercept processing occurred as you expected for DB2 utilities by checking the DB2 UET messages that are incorporated into the SYSPRINT data set for the utility job and the SYSPRINT data set for the DB2 UET started task. Use SDSF or an equivalent tool to view this information.

Important: These messages pertain to all types of DSNUTILB intercept processing. You should also check the output that is specific to the intercept features that you implemented, as described in the following topics:

- "Determining whether the LOAD utility enhancements were implemented" on page 288
- "Determining whether the REORG TABLESPACE utility enhancements were implemented" on page 293
- "Determining whether thread blocking and canceling occurred" on page 184

Messages in the SYSPRINT data set for a DB2 utility

The following table explains the key DB2 UET messages on DSNUTILB interception that can occur in the SYSPRINT data for a DB2 utility. Look for these messages to determine if interception processing completed as you intended. The messages are described in the order in which they appear in the SYSPRINT data set.

Messages that are issued for a worklist step (a utility command) are often paired; the first message provides the step number of the worklist step, and the second message provides the return code for that worklist step. A return code of less than 8 is ignored; DSNUTILB intercept processing continues. A return code of 8 or higher indicates that an error occurred and DSNUTILB intercept processing terminated abnormally. The return codes in messages that pertain to thread-cancelation processing can be from either the DSNUTILB intercept or the batch interface. The intercept calls the batch interface during intercept processing.

Table 30. Intercept messages in the utility SYSPRINT data set

Messages	Explanation
ABPU5001I *date time* IBM DB2 Utilities Enhancement Tool Version *product_version*, FMID=*product_fmid*, COMP_ID=*product_compid*	The specified version of the product is installed and the DSNUTILF module, which is required for intercept processing, is available.
ABPU5012I *date time* Connected to started task ABPID=*configuration_id* ABPU5002I *date time* Initialization is complete.	The DSNUTILB intercept connected to the specified DB2 UET started task configuration and completed initialization.
ABPU5340E *date time* Worklist in use by another utility ID=*utility_ID*	DSNUTILB interception cannot occur because a worklist for the specified utility ID already exists and is currently in use by another utility job. In this case, refer to the ABPS5113I message in the SYSPRINT data set for the started task for more information.
ABPU5004I *date time* Analysis started. Step=*step_number* ABPU5005I *date time* Analysis completed. RC=*return_code*	DB2 UET began the analysis phase for the specified worklist step and then completed the analysis phase with the specified return code. This return code is issued from the DSNUTILB intercept.

Table 30. Intercept messages in the utility SYSPRINT data set (continued)

Messages	Explanation
ABPU5008I *date time* Utility execution started. Step=*step_number* ABPU5009I *date time* Utility execution completed. RC=*return_code*	The DB2 utility command that is associated with the specified worklist step began execution. The utility command then completed execution with the specified return code. These messages are issued for each utility command that is in the original DSNUTILB SYSIN stream. The return code in ABPU5009I is from either the DB2 utility or the DSNUTILB intercept. The intercept return code is used if: 1) it is 8 or greater and 2) it is equal to or greater than the utility return code. The highest return code that is provided in any ABPU5009I message for a worklist step will be the return code for the entire utility job.
ABPU5003I *date time* DB2 Utilities Enhancement Tool intercept completed.	The DSNUTILB intercept completed intercept processing for the utility.

For utility enhancements that modify the original DSNUTILB SYSIN stream (the additional options for the LOAD utility and the mapping-table enhancement for the REORG TABLESPACE utility), messages ABPU5330, ABPU5331, and ABPU5332 are also written to the SYSPRINT data set to present the enhanced DSNUTILB SYSIN stream. To determine whether the SYSIN was correctly processed, compare this SYSIN stream for the utility with the subsequent DSNUTILB messages.

Messages in the SYSPRINT data set for the started task

The following table explains the key messages on DSNUTILB intercept processing that can occur in the SYSPRINT data set for the started task.

Table 31. Intercept messages in the started task SYSPRINT data set

Messages	Explanation
ABPS0101I *date time* TCB: *tcb_address* Session created. SESS:*session_token-session_number-session_type-session_job_name-session_job_ID-session_asid-session_user*	A DB2 UET session was created for DSNUTILB intercept processing. Sessions for the DSNUTILB intercept have a session type of "U."

Table 31. Intercept messages in the started task SYSPRINT data set (continued)

Messages	Explanation
ABPS5100I *date time* TCB: *tcb_address* Session: *session_token* SSID: *ssid* DSNUTILB utility id : *utility_id* *message_continuation_number* ABPS5101I *date time* *message_continuation_number* DSNUTILB intercept operation is *operation_name* ABPS5113I *date time* *message_continuation_number* Worklist is in use by another utility. Owning Session: *session_token* ABPS5111E *date time* *message_continuation_number* DSNUTILB intercept operation failed	If an error occurred during DSNUTILB processing for a utility ID, the message ABPS5111E is issued along with the messages ABPS5100I and ABPS5101I, which provide more information about the intercept operation. If the failure occurred because the worklist is already in use under the same utility ID, the ABPS5113I message is also issued. In this case, refer to the preceding ABPS0101I message that contains a matching session token value to determine the job name and job ID of the utility job that is currently using the worklist.

Related concepts:

"Determining whether the LOAD utility enhancements were implemented" on page 288
To determine whether the DB2 UET options for the LOAD utility were processed, check the DB2 UET messages in the SYSPRINT data set for the LOAD utility. Use SDSF or an equivalent tool to view this information.

"Determining whether the REORG TABLESPACE utility enhancements were implemented" on page 293
To determine whether DB2 UET was able to generate the mapping table and mapping table index for the REORG TABLESPACE utility, check the DB2 messages in the SYSPRINT data set for the utility. The DB2 UET SYSPRINT displays message ABPU5407I to indicate that the mapping table was created successfully.

"Determining whether thread blocking and canceling occurred" on page 184
To determine whether thread blocking and cancelation processing occurred, you can check the DB2 UET messages in the SYSPRINT data set for the DB2 utility. Also, you can review the batch reports on threads canceled. Use SDSF or an equivalent tool to view this information.

Deactivating DSNUTILB intercept processing

You stop DB2 UET DSNUTILB intercept processing temporarily by issuing a z/OS console command. You can resume intercept processing later by issuing another console command.

About this task

- Variable *abpstc* is the member name of the DB2 UET PROC in the system PROCLIB.
- Omit the **GLOBAL** parameter to deactivate interception for the specified started task only.

Procedure

To deactivate DSNUTILB interception across the entire z/OS image, (rather than the specified started task only), from the z/OS console, issue one of the following MODIFY commands:

- `F abpstc,DEACTIVATE INTERCEPT[,GLOBAL]`

- If you use SDSF:

  ```
  /F abpstc,DEACTIVATE INTERCEPT[,GLOBAL]
  ```

Results

After DSNUTILB intercept processing is deactivated, any intercept processing that you configured for the DB2 utility enhancements or thread blocking/canceling will not occur.

What to do next

To resume DSNUTILB intercept processing, you must reactivate the intercept by issuing the **ACTIVATE INTERCEPT** console command. For more information, see "Reactivating the DSNUTILB intercept."

Related tasks:

"Reactivating the DSNUTILB intercept"
By default, DSNUTILB interception is enabled globally across the z/OS image on which DB2 UET is running. However, if you deactivate the intercept, you will need to reactivate it to resume intercept processing.

Reactivating the DSNUTILB intercept

By default, DSNUTILB interception is enabled globally across the z/OS image on which DB2 UET is running. However, if you deactivate the intercept, you will need to reactivate it to resume intercept processing.

Before you begin

Before you reactivate the intercept, ensure that the DSNUTILB intercept policy is still valid and that the name of the policy member is specified in the ABPPLCY DD statement of the started task PROC.

About this task

- Variable *abpstc* is the member name of the DB2 UET PROC in the system PROCLIB.
- Include the **GLOBAL** parameter if you previously deactivated DSNUTILB interception across the entire z/OS image by issuing the **ACTIVATE INTERCEPT,GLOBAL** command and now want to reactivate interception across the entire z/OS image.

Procedure

To reactivate the intercept, from the z/OS console, issue one of the following MODIFY commands:

- `F abpstc,ACTIVATE INTERCEPT[,GLOBAL]`
- If you use SDSF:

  ```
  /F abpstc,ACTIVATE INTERCEPT[,GLOBAL]
  ```

Results

After the DSNUTILB intercept is reactivated, it can once again intercept DSNUTILB to implement any intercept processing that you configured for the DB2 utilities (that is, the additional DB2 utility options or intercept processing).

Related concepts:

"Task flow for blocking and canceling threads" on page 183
This task flow presents the high-level tasks that you will need to perform to block and cancel threads by using the DSNUTILB intercept.

Related tasks:

"Deactivating DSNUTILB intercept processing" on page 312
You stop DB2 UET DSNUTILB intercept processing temporarily by issuing a z/OS console command. You can resume intercept processing later by issuing another console command.

Displaying the DSNUTILB intercept status

You can write the DSNUTILB intercept status (Enabled or Disabled) to the SYSPRINT data set that is allocated to the started task by specifying a z/OS console command. If interception is not occurring as expected, you might want to check the intercept status to determine if another user deactivated the intercept.

About this task

- You can control the DSNUTILB intercept status by issuing the **DEACTIVATE INTERCEPT** or **ACTIVATE INTERCEPT** console command.
- Variable *abpstc* is the member name of the DB2 UET PROC in the system PROCLIB.
- The **GLOBAL** and **ALL** parameters are optional, and can be used as follows:
  - Without the optional parameters, the command displays the local status that is set for the specified started task only.
  - If you specify the **GLOBAL** parameter, the command displays the global interception status that is set for the entire z/OS image. (You control the global status by issuing the **DEACTIVATE INTERCEPT,GLOBAL** or **ACTIVATE INTERCEPT,GLOBAL** command.)
  - If you specify the **ALL** parameter, the command writes all of the following information to the SYSPRINT data set: the local interception status; the global interception status; and a list of the DB2 SSIDs for which DSNUTILB interception is occurring, including the ABPID (configuration ID) of the started task that is involved in intercept processing.

Procedure

1. To display the current intercept status, from the z/OS console, issue one of the following MODIFY commands:
   - `F abpstc,DISPLAY INTERCEPT[,GLOBAL|ALL]`
   - If you use SDSF:

     `/F abpstc,DISPLAY INTERCEPT[,GLOBAL|ALL]`
2. After issuing the command, navigate to the SYSPRINT data set for the started task to view the command output.

Example

The following example displays the messages that resulted from the **DISPLAY INTERCEPT,ALL** command. These messages indicate the local intercept status, the global intercept status, and the SSID of the single subsystem for which DSNUTILB interception is occurring.

```
ABPS0814I date_timestamp Command issued: DISPLAY INTERCEPT,ALL
ABPS0817I date_timestamp LOCAL  DSNUTILB intercept status is: ENABLED
ABPS0817I date_timestamp GLOBAL DSNUTILB intercept status is: ENABLED
ABPS0822I date_timestamp DB2 SSID=DBP1 810 ABPID=ABP01 DSNUTILB interception is installed
```

Displaying the current intercept policy

You can write the contents of the current DSNUTILB intercept policy to the SYSPRINT data set that is allocated to the DB2 UET started task by specifying a z/OS console command. This command provides a quick way to check the DB2 SSIDs and rules that the policy specifies.

About this task

Variable *abpstc* is the member name of the DB2 UET PROC in the system PROCLIB.

Procedure

1. To display the policy contents, from the z/OS console, issue one of the following MODIFY commands:
   - `F abpstc,DISPLAY POLICY`
   - If you use SDSF:

     `/F abpstc,DISPLAY POLICY`
2. Navigate to the SYSPRINT data set for the started task to view the output from the DISPLAY POLICY command. This output is composed of a list of the DB2 SSIDs that are specified in the policy and any INCLUDE or EXCLUDE rules that are defined for each SSID.

Example

The following example output is for a simple policy that contains one inclusion rule for the DBP1 subsystem:

```
Command issued: DISPLAY POLICY
DSNUTILB Intercept Policy:
 DB2 SSID: DBP1
  Rule type: INCLUDE
   00001:UTILITY_USERID                  = DBA%
```

Interception problems for thread cancelations

If DSNUTILB interception fails or yields unexpected results for a thread cancelation, you can check the output in the ABPRDIAG data set that is allocated to the DB2 UET started task. This data set indicates whether interception was enabled or disabled based on each INCLUDE rule and EXCLUDE rule in the policy.

For each INCLUDE rule and EXCLUDE rule, the ABPRDIAG data set shows:

- The value that is specified in the policy (P value).
- The actual field value (F value) against which the policy value was matched.
- Whether the P value matched the F value.
- Whether interception was set to On or Off when a match occurred. Interception is set to On when the P value in an INCLUDE rule matches an F value and is set to Off when the P value in an EXCLUDE rule matches an F value. (For no-match situations, the intercept status remains unchanged.)

The session number and rule numbers that are specified in the ABPRDIAG data set correspond to the session number and rule numbers that are displayed in the SYSPRINT data set for the started task. You can refer to SYSPRINT data set for details on the policy rules.

Review the following examples to learn how to analyze interception problems related to thread cancelation. If you need to contact IBM Software Support, see "Diagnostic information for Support" on page 439.

Example 1

DB2 utility statement: Assume that DSNUTILB issued the following LOAD utility statement (only the part of the statement that applies to this example is shown) when you use the DSNUTILB intercept to perform thread blocking and cancelation for the utility:

```
LOAD DATA REPLACE
   INTO TABLE ABPUDB01.US01TB01
         (specifications... )
   INTO TABLE ABPUDB01.US01TB02
         (specifications... )
   INTO TABLE ABPXDB01.US01TB03
         (specifications... )
```

SYSPRINT information: The SYSPRINT data set for the started task contains the following policy information:

```
ABPS0830I date_timestamp DSNUTILB Intercept Policy:
ABPS0831I date_timestamp  DB2 SSID: DBP1
ABPS0832I date_timestamp   Rule type: INCLUDE
ABPS0833I date_timestamp    00001:UTILITY_USERID             = PDABCD%
ABPS0833I date_timestamp    00002:UTILITY_C                  = LOAD
ABPS0833I date_timestamp    00003:TABLE                      = ABPUDB0%.U%
```

The rule numbers are shown in bold. Rule 3 for the TABLE attribute has a value in the format *creator_id.table_name*.

The SYSPRINT data set also indicates the session number (00000001) as follows:

```
ABPS0101I date_timestamp TCB: 008CF6D8 Session created.
SESS:24EE30B0-00000001-B-ABPUTEST-J0123456-00CC-PDABCD
```

ABPRDIAG information: The corresponding ABPRDIAG data set for the started task presents the following results from the matching of the policy rules against the actual field values. (The session number and rule numbers correspond to those in the SYSPRINT data set.)

```
SESSION: 00000001    PHASE: 00000002    STEP: 00000001    C: LOAD
  INCLUDE  UTILITY_USERID       RULE NUMBER: 00000001  MATCH    INTERCEPT ON
    P: PDABCD%
    F: PDABCD
  INCLUDE  UTILITY_C          RULE NUMBER: 00000002  MATCH    INTERCEPT ON
    P: LOAD
    F: LOAD
  INCLUDE  TABLE                RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: ABPUDB0%
    F: ABPUDB01
  INCLUDE  TABLE                RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: U%
    F: US01TB01
  INCLUDE  TABLE                RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: ABPUDB0%
    F: ABPUDB01
  INCLUDE  TABLE                RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: U%
    F: US01TB02
  INCLUDE  TABLE                RULE NUMBER: 00000003  NOMATCH  INTERCEPT OFF
    P: ABPUDB0%
    F: ABPXDB01
```

Analysis: The policy contains three types of INCLUDE rules (for the UTILITY_USERID, UTILITY_C, and TABLE attributes). Each of these rules must match the LOAD input for thread blocking and cancelation to occur. For the TABLE rule (rule 3), each part of the *creator_id.table_name* value is matched separately against the actual creator IDs and table names of the tables to be loaded.

The matching results in the ABPRDIAG data set indicate that each type of INCLUDE rule passed the matching test in full or in part. The first three matches for the INCLUDE TABLE rule (rule 3) succeeded but the last match failed. For thread blocking and cancelation to occur, all of the INCLUDE TABLE matches must result in setting INTERCEPT ON for the tables to be loaded. However, for the ABPXDB01.US01TB03 table, the matching of the ABPXDB01 value (F value) against the ABPUDB0% value in the policy (P value) resulted in setting INTERCEPT OFF. Consequently, no thread blocking and cancelation will occur for the entire LOAD operation.

Example 2

DB2 utility statement: Assume that DSNUTILB issued the following LOAD utility statement (only the part of the statement that applies to this example is shown) when you are using the DSNUTILB intercept to perform thread blocking and cancelation for the utility:

```
LOAD DATA REPLACE
   INTO TABLE ABPUDB01.US01TB01
         (specifications... )
   INTO TABLE ABPUDB01.US01TB02
         (specifications... )
```

SYSPRINT information: The SYSPRINT data set for the started task contains the following policy information:

```
ABPS0830I date_timestamp DSNUTILB Intercept Policy:
ABPS0831I date_timestamp  DB2 SSID: DBP1
ABPS0832I date_timestamp   Rule type: INCLUDE
ABPS0833I date_timestamp    00001:UTILITY_USERID        = PD%
ABPS0833I date_timestamp    00002:UTILITY_C             = LOAD
ABPS0833I date_timestamp    00003:TABLE                 = ABPUDB0%.U%
ABPS0832I date_timestamp   Rule type: EXCLUDE
ABPS0833I date_timestamp    00004:UTILITY_USERID        = PDINT%
```

ABPRDIAG information: The corresponding ABPRDIAG data set for the started task presents the following results from matching the policy rules against actual field values.

```
SESSION: 00000002    PHASE: 00000002    STEP: 00000001    C: LOAD
  INCLUDE  UTILITY_USERID           RULE NUMBER: 00000001  MATCH    INTERCEPT ON
    P: PD%
    F: PDINTRN1
  INCLUDE  UTILITY_C                RULE NUMBER: 00000002  MATCH    INTERCEPT ON
    P: LOAD
    F: LOAD
  INCLUDE  TABLE                    RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: ABPUDB0%
    F: ABPUDB01
  INCLUDE  TABLE                    RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: U%
    F: US01TB01
  INCLUDE  TABLE                    RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: ABPUDB0%
    F: ABPUDB01
  INCLUDE  TABLE                    RULE NUMBER: 00000003  MATCH    INTERCEPT ON
    P: U%
```

```
  F: US01TB02
EXCLUDE  UTILITY_USERID           RULE NUMBER: 00000004  MATCH    INTERCEPT OFF
  P: PDINT%
  F: PDINTRN1
```

Analysis: The policy contains three types of INCLUDE rules (for the UTILITY_USERID, UTILITY_C, and TABLE attributes), each of which must match at least one actual field value for interception to occur. The policy also contains one EXCLUDE rule for the UTILITY_USERID attribute (rule 4). Because rules are processed from top to bottom, rule 4 is processed after the INCLUDE rule for the UTILITY_USERID (rule 1).

The matching results in the ABPRDIAG data set indicate that all three types of INCLUDE rules matched actual data. Both the INCLUDE UTILITY_USERID rule (rule 1) and EXCLUDE UTILITY_USERID rule (rule 4) matched the actual user ID of "PDINTRN1." Because rule 4 is processed after rule 1, rule 4 overrides rule 1 and excludes that user ID from interception. As a result, interception does not occur because the only user ID that met the UTILITY_USERID inclusion criterion was subsequently excluded.

Terminating a DB2 utility for which interception has occurred

If you need to terminate a DB2 utility for which DSNUTILB intercept processing is occurring or has occurred, use the ABPMAINT utility that is provided by DB2 UET.

About this task

The ABPMAINT utility both issues the DB2 -TERM UTILITY command for a specific utility ID and removes the worklist rows that contain that utility ID from all intercept worklist tables. If you run the ABPMAINT utility for a thread blocking and cancelation operation after threads are blocked but before the allow-threads step has taken place, the ABPMAINT utility also changes the access status of the DB2 objects to allow threads to form again.

If you manually issue the DB2 -TERM UTILITY command instead, also run the ABPMAINT utility to remove the data for the terminated utility (utility ID) from the worklist tables. If the data for the terminated utility remains in the worklist tables and you restart the utility, the DSNUTILB intercept attempts to resume utility processing from the beginning of the current worklist step, as identified in the worklist tables.

Procedure

1. Ensure that the DB2 UET started task is running and that the DB2 plan for DB2 UET is bound on the subsystem against which the DB2 utility is running.
2. Customize the JCL for the ABPMAINT utility, which is located in the *abpid*MNT member (where *abpid* is the DB2 UET configuration ID) in the *hlq.mlq*.SABPSAMP library, as follows:

   Remember: The Tools Customizer creates a separate *abpid*MNT member for each started task configuration that you define.

   a. Add a job card, if necessary. If you specified a job card template when you ran the Tools Customizer, that job card information should already exist.
   b. In the EXEC statement, set the following options on the PARM:

      ```
      PARM='abpid,#FUNCTION#,#DB2SSID#,#UTILITY_ID#'
      ```

      Where:

- *abpid* is the configuration ID of the DB2 UET started task that you are using to perform DSNUTILB interception. You specified this value in Tools Customizer during customization. Tools Customizer should have inserted this value for you.
- #FUNCTION# must be the value TERM_UTILITY (the name of the ABPMAINT function for terminating a DB2 utility and cleaning up the worklist tables).
- #DB2SSID# represents the subsystem identifier (SSID) of the DB2 subsystem against which the DB2 utility is running.
- #UTILITY_ID# represents the DB2 identifier (UTILID) for the DB2 utility.

c. In the STEPLIB DD statement, specify the high-level qualifier (?HLQ?) and the mid-level qualifier (?MLQ?) for the ABPLOAD library, if necessary. The Tools Customizer should have inserted these values for you.

3. Submit the ABPMAINT job.

Results

The ABPMAINT utility terminates the DB2 utility and removes all data that is associated with the utility ID from the worklist tables. If thread blocking occurred, the ABPMAINT utility might also change the status of the DB2 objects to allow threads to form again.

Related reference:

"Validating records before committing changes (IFDISCARDS option)" on page 281
The IFDISCARDS option for the DB2 LOAD utility provides Load Prevalidate feature.

Restarting a DB2 utility in exceptional circumstances

When a DB2 utility for which DSNUTILB interception is occurring terminates abnormally, DB2 can usually resume utility processing from the appropriate point, without any special user intervention, when you restart the utility. However, in some exceptional circumstances, a normal DB2 restart of a DB2 utility might not be possible. In these circumstances, you can use the DB2 UET ABPMAINT utility to resume utility processing.

About this task

Consider using the ABPMAINT utility for restart purposes when an event such as an abend of the DB2 UET started task or of DB2 occurs and causes the DB2 utility to end before DB2 UET has recorded the status of the last utility-command operation within a worklist step in the intercept worklist tables. In this situation, you can use the ABPMAINT utility to resume utility processing from the last utility-command operation in the current worklist step, from the next operation within the current worklist step, or from the next worklist step. For more information, see "How the DSNUTILB intercept affects the restart of DB2 utilities" on page 128.

Procedure

1. Ensure that the DB2 UET started task is running and that the DB2 plan for DB2 UET is bound on the subsystem against which the DB2 utility is running.
2. Customize the JCL for the ABPMAINT utility, which is located in the *abpid*MNT member (where *abpid* is the DB2 UET configuration ID) in the *hlq.mlq*.SABPSAMP library, as follows:

Remember: Tools Customizer creates a separate *abpid*MNT member for each started task configuration that you define.

a. Add a job card, if necessary. If you specified a job card template when you ran the Tools Customizer, that job card information should already exist.

b. In the EXEC statement, set the following options on the PARM:

```
PARM='abpid,#FUNCTION#,#DB2SSID#,#UTILITY_ID#'
```

Where:

- *abpid* is the configuration ID of the DB2 UET started task that you are using to perform DSNUTILB interception. You specified this value during customization.
- #FUNCTION# must be one of the following literal values, which identifies the restart function you want to use:
  - FORCE_RESTART - Sets the status of the last utility-command operation within the current worklist step (the operation for which the status was not recorded when the utility ended) to force the utility to restart from that operation.
  - MARK_COMPLETE - Sets the status of the last utility-command operation within the current worklist step (the operation that completed but was not recorded as complete when the utility ended) to complete. DB2 UET assumes that the utility-command operation was successful. When you restart the DB2 utility, it resumes intercept processing from the beginning of the next operation in the current worklist step.
  - STEP_ADVANCE - Sets the status of the current worklist step to complete. When you restart the DB2 utility, it resumes intercept processing from the beginning of the next worklist step. Specify this function only if you are prepared to manually perform any required operations that the intercept did not finish for the current worklist step before the status of that worklist step was set to complete. For example, you might have to drop the mapping tables and mapping-table indexes that were created for the REORG TABLESPACE utility or to reset the access statuses of DB2 objects for which threads were blocked.
  - TERM_UTILITY - Terminates the utility instead of restarting it. For more information, see "Terminating a DB2 utility for which interception has occurred" on page 318.
- #DB2SSID# represents the subsystem identifier (SSID) of the DB2 subsystem against which the DB2 utility is running.
- #UTILITY_ID# represents the DB2 identifier (UTILID) for the DB2 utility.

c. In the STEPLIB DD statement, specify the high-level qualifier (?HLQ?) and the mid-level qualifier (?MLQ?) for the ABPLOAD library, if necessary. The Tools Customizer should have inserted these values for you.

3. Submit the ABPMAINT job.
4. When the ABPMAINT job is completed, restart the DB2 utility. The utility resumes processing based on the ABPMAINT function that you specified.

Related concepts:

"How the DSNUTILB intercept affects the restart of DB2 utilities" on page 128
The DSNUTILB intercept supports the normal restart capabilities of all DB2 versions that DB2 UET supports. When a DB2 utility for which interception is occurring terminates abnormally and then you restart it, DB2 resumes utility processing from the appropriate point without any special user intervention.

Chapter 12. Troubleshooting

Use these topics to diagnose and correct problems that you experience with DB2 Utilities Enhancement Tool.

Topics:

Tools Customizer troubleshooting

Use this information to diagnose and correct problems that you experience with Tools Customizer.

Gathering diagnostic information

Before you report a problem with Tools Customizer to IBM Software Support, you need to gather the appropriate diagnostic information.

Procedure

Provide the following information for all Tools Customizer problems:

- A clear description of the problem and the steps that are required to re-create the problem
- Relevant screen captures
- All messages that were issued as a result of the problem
- Product release number and the number of the last program temporary fix (PTF) that was installed
- The version of DB2 that you are using and the type and version of the operating system that you are using
- The Tools Customizer trace data set
- The Tools Customizer data store data set and the *high_level_qualifier*.SCCQTENU data set

Determining the trace data set name

You will need to identify the name of the trace data set if you cannot allocate the trace data set, the trace data set runs out of space, or IBM Software Support asks for it.

The name of the trace data set depends on the prefix setting in the TSO profile. To identify the name of the trace data set, you must know the prefix setting.

- If PREFIX is set, the name of the trace data set is *prefix*.CCQ.TRACE, where *prefix* is the TSO prefix that you specified in the profile.
- If NOPREFIX is set, the name of the trace data set is *user_ID*.CCQ.TRACE, where *user_ID* is your TSO user ID.

How to look up message explanations

You can use several methods to search for messages and codes.

Searching an information center

In the search box that is located in the top left toolbar of any Eclipse help system, such as the IBM Information Management Software for z/OS Solutions Information Center, enter the number of the message that you want to locate. For example, you can enter DFS1065A in the search field.

Use the following tips to help you improve your message searches:

- You can search for information on codes by entering the code; for example, enter -327.
- Enter the complete or partial message number. You can use the asterisk wildcard character (*) to represent multiple characters, and you can use the question mark wildcard character (?) to represent a single character.

The information center contains the latest message information for all of the information management products that are included in the information center.

Searching for messages on the Web

You can use any of the popular search engines that are available on the Web to search for message explanations. When you type the specific message number or code into the search engine, you will be presented with links to the message information in IBM information centers.

Using LookAt

LookAt is an online facility that you can use to look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA, and Clusters for AIX® and Linux:

- The Internet. You can access IBM message explanations directly from the LookAt website at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, a TSO/E prompt, ISPF, or z/OS UNIX System Services running OMVS).
- Your Microsoft Windows workstation. You can install code to access IBM message explanations on the z/OS Collection (SK3T-4271) using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt website.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your z/OS Collection (SK3T-4271) or from the LookAt website (click **Download** and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Messages and codes

These topics contain information about the messages, return codes, and abend codes that can be issued by DB2 UET.

Topics:

Tools Customizer messages

Use the information in these messages to help you diagnose and solve Tools Customizer problems.

CCQB000I The product parameter data was saved in the data store.

Explanation: Changes that were made to the product parameters were saved in the data store.

System action: None.

User response: No action is required.

CCQB001I The DB2 parameter data was saved in the data store.

Explanation: Changes that were made to the DB2 parameters were saved in the data store.

System action: None.

User response: No action is required.

CCQB002I The LPAR parameter data was saved in the data store.

Explanation: Changes that were made to the LPAR parameters were saved in the data store.

System action: None.

User response: No action is required.

CCQB003E At least one step must be selected in a selected task. The selected task is *task_description***.**

Explanation: When a task is selected, at least one step must be selected. A selected step is missing from the specified task.

System action: Processing stops.

User response: Select a step in the specified task or deselect the task.

CCQB004I The required information to run the Discover EXEC was saved in the data store.

Explanation: The data store contains all the information that is required to run the Discover EXEC.

System action: None.

User response: No action is required.

CCQB005E The conflicting values for the *parameter_name* **parameter must be resolved before the information can be saved.**

Explanation: Two values for one parameter conflict with each other, and they must be resolved to save the information.

System action: Processing stops.

User response: Resolve the conflicting values for the parameter.

CCQB006E One row must be selected.

Explanation: One row in the table must be selected.

System action: Processing stops.

User response: Select one row.

CCQB007E Only one row can be selected.

Explanation: Multiple rows in the table are selected, but only one row is allowed to be selected.

System action: Processing stops.

User response: Select only one row.

CCQC000I **The jobs have been customized on the selected DB2 entries.**

Explanation: The jobs were customized on the DB2 entries that were selected.

System action: None.

User response: Press Enter to clear the message.

CCQC001W **The jobs were not generated on one or more of the selected DB2 entries. Press PF3 to check the DB2 entries that were not customized.**

Explanation: The product was not customized on one or more of the DB2 entries that were selected.

System action: None.

User response: Press PF3 to see the DB2 entries on which the product was not customized. The status of these DB2 entries is Errors in Customization.

CCQC002I **The edit session was started automatically because values for required parameters are missing or must be verified.**

Explanation: If product, LPAR parameters, or DB2 parameters are not defined or if parameter definitions must be verified, an editing session for the undefined or unverified parameters starts automatically.

System action: None.

User response: Define values for all required product, LPAR parameters, or DB2 parameters.

CCQC003W **The** *template_name* **template in the** *library_name* **metadata library does not contain any parameters.**

Explanation: The specified template does not have parameters.

System action: None.

User response: No action is required.

CCQC004S **The value of the "type" attribute for the** *template_name* **template in the** *library_name* **metadata library does not match the value that was previously specified. The value is** *value_name***, and the previously specified value is** *value_name***.**

Explanation: The value of the "type" attribute must match the value that was previously specified.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC005S **The** *template_name* **template exceeds the number of allowed templates for a customization sequence. The template is in the** *library_name* **metadata library.**

Explanation: The customization sequence can process only *number* templates. The specified template cannot be processed because the customization sequence already contains the maximum number of templates.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC006E **The jobs could not be generated for the** *group_attach_name* **DB2 group attach name.**

Explanation: The customization jobs could not be generated for the specified DB2 group attach name.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC007E **The jobs could not be generated for the** *subsystem_ID* **DB2 subsystem.**

Explanation: The customization jobs could not be generated for the specified DB2 subsystem.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC008E **The jobs could not be generated for the** *member_name* **DB2 member.**

Explanation: The customization jobs could not be generated for the specified DB2 member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC009S **The jobs were not generated for the DB2 entries.**

Explanation: One or more errors occurred while customization jobs were being generated for the selected DB2 entries.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC010S **The** *template_name* **template could not be accessed in the** *library_name* **metadata library.**

Explanation: The specified template could not be accessed because the user does not have RACF access

to the data set, the data set has incorrect data characteristics, or the data set is not cataloged.

System action: Processing stops.

User response: Ensure that you have RACF access to the data set, that the characteristics are correct according to the specifications of the product that you are customizing, and that the data set is cataloged. If the problem persists, contact IBM Software Support.

CCQC011S **The *template_name* template could not be written to the *library_name* customization library.**

Explanation: The specified template could not be accessed because the user does not have RACF access to the data set, the data set has incorrect data characteristics, or the data set is not cataloged.

System action: Processing stops.

User response: Ensure that you have RACF access to the data set, that the characteristics are correct according to the specifications of the product that you are customizing, and that the data set is cataloged. If the problem persists, contact IBM Software Support.

CCQC012W **The job card was generated with default values because the JOB keyword was missing.**

Explanation: Default values were used to generate the job card because the JOB keyword was not specified in the first line of the job card.

System action: The job card was generated with default values.

User response: No action is required. To generate the job card with your own values, add the JOB keyword in the first line of the job card.

CCQC013W **The job card was generated with the default value for the programmer name because the specified programmer name exceeded 20 characters.**

Explanation: Default values were used to generate the job card because the specified programmer name contained too many characters.

System action: The job card was generated with default values.

User response: No action is required. To generate the job card with your own values, add a valid programmer name in the job card. A valid programmer name is 1 - 20 characters.

CCQC014W **The job card was generated with default values because the JOB keyword was not followed by a space.**

Explanation: Default values were used to generate the job card because a space did not follow the JOB keyword.

System action: The job card was generated with default values.

User response: No action is required. To generate the job card with your own values, add a space after the JOB keyword in the job card.

CCQC015S **The *template_name* template in the *library_name* metadata library contains the following file-tailoring control statement: *statement_name*. This control statement is not valid in a *template_type* template.**

Explanation: The *template_type* template cannot contain the specified type of file-tailoring control statement.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC016S **The)DOT file-tailoring control statement exceeded the number of allowed occurrences for the *template_name* template in the *library_name* metadata library.**

Explanation: The)DOT file-tailoring control statement can occur only a limited number of times in the specified template.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC017S **The nested)DOT file-tailoring control statements exceeded the number of allowed occurrences in the *template_name* template in the *library_name* metadata library.**

Explanation: Nested)DOT file-tailoring control statements can occur only *number* times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC018S **The *template_name* template in the *library_name* metadata library is not valid because it does not contain any data.**

Explanation: The specified template is missing required data.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC019S **The *template_name* template in the *library_name* metadata library is not valid because an)ENDDOT file-tailoring control statement is missing.**

Explanation: A)ENDDOT file-tailoring control statement is required in the specified template.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC021S **The *template_name* template in the *library_name* metadata library is not valid because the template must start with the *parameter_name* job card parameter.**

Explanation: The specified template must start with the specified job card parameter.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC022S **The parameters used in a)DOT file-tailoring control statement exceeded the number of allowed parameters in the *template_name* template. The template is in the *library_name* metadata library. The error occurs in)DOT section *section_number*.**

Explanation: A)DOT file-tailoring control statement can contain only a limited number of parameters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC023S **The)DOT file-tailoring control statement must include the *table-name* table name in the *template_name* template. The template is in the *library_name* metadata library. The error occurs in)DOT section *section_number*.**

Explanation: The)DOT file-tailoring control statement is missing a required table name.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC024S **ISPF file tailoring failed for the *template_name* template in the *library_name* metadata library.**

Explanation: An error occurred during ISPF file tailoring for the specified template.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC025I **Customized jobs do not exist because they have not been generated.**

Explanation: The list of customized jobs cannot be displayed because the product has not been customized for any DB2 entries.

System action: None.

User response: Complete the steps to customize a product. Customized jobs are generated when all required product, LPAR parameters, and DB2 parameters are defined and at least one DB2 entry on which to customize the product has been selected.

CCQC026S **The value of the "customized" attribute for the *parameter_name* parameter in the *library_name* metadata library template does not match the value that was previously specified. The value is *value_name*, and the previously specified value is *value_name*.**

Explanation: The value for the "customized" attribute for a parameter must match the value that was previously specified.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC027S **The *job_name* customization job was not found in the *library_name* customization library.**

Explanation: The selected customization job does not exist in the customization library.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC028S **The *library_name* customization library was not found.**

Explanation: The customization library does not exist.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQC029I **The customization jobs were generated for *Product_name*.**

Explanation: The customization jobs were generated for the specific product.

System action: None.

User response: No action is required.

CCQC030S The customization jobs cannot be generated because at least one DB2 entry must be associated with this product.

Explanation: The product that you are customizing requires at least one DB2 entry to be associated with it before customization jobs can be generated.

System action: None.

User response: Associate a DB2 entry with the product that you are customizing, and regenerate the jobs.

CCQC031I The jobs were generated for the associated DB2 entries.

Explanation: The customization jobs were generated for the DB2 entries that are associated with the product.

System action: None.

User response: No action is required.

CCQC032S The customization jobs were not generated for *Product_name***.**

Explanation: A severe error occurred while the jobs were being generated for the specified product.

System action: None.

User response: Contact IBM Software Support.

CCQC033S The *customization_library_name* **has no customized jobs.**

Explanation: The specified customization library cannot be browsed or edited because it is empty.

System action: None.

User response: Generate customization jobs for the specified library, and browse or edit the library again.

CCQC034S The specified operation is not allowed.

Explanation: Issuing commands against customization jobs from the customization library from an ISPF browse or edit session that was started on the Finish Product Customization panel is restricted.

System action: None.

User response: To make changes to customization jobs, follow the steps for recustomization.

CCQC035E Before you generate customization jobs, edit the product parameters to select one or more tasks or steps, and then issue the G line command or the GENERATEALL command again.

Explanation: One or more tasks or steps must be selected before customization jobs can be generated.

System action: None.

User response: Edit the product parameters to select one or more tasks or steps. Then, issue the G line command or the GENERATEALL command again.

CCQC036E Before you exit the Product Parameters panel, you must select one or more tasks or steps to generate customization jobs or issue the CANCEL command.

Explanation: One or more tasks or steps must be selected to generate customization jobs or the CANCEL command must be issued before you can exit the Product Parameters panel.

System action: None.

User response: Select one or more tasks or steps, or issue the CANCEL command.

CCQD000W The *member_name* **environment index member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the specified environment index member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the warning.

CCQD001S The *member_name* **environment index member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the specified environment index member is valid, the PL/I XML parser issued an exception error code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the error.

CCQD002S The XML structure of the *member_name* **environment index member is not valid. The** *element_name* **element is unknown.**

Explanation: The specified environment index member contains an unknown element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD003S **The XML structure of the *member_name* environment index member is not valid. Content is not allowed for the *element_name* element, but content was found.**

Explanation: Content was found in an element that cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD004S **The XML structure of the *member_name* environment index member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD005S **The XML structure of the *member_name* environment index member is not valid. The content length for the *element_name* element exceeds *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD006S **The XML structure of the *member_name* environment index member is not valid. The *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified element occurs too many times in the environment index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD007S **The XML structure of the *member_name* environment index member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times in the environment index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD008S **The XML structure of the *member_name* environment index member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times in the environment index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD009S **The XML structure of the *member_name* environment index member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute does not occur enough times in the environment index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD010S **The XML structure of the *member_name* environment index member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: Content was found in an attribute that cannot contain content. The name of the attribute and the name of the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD011S **The XML structure of the *member_name* environment index member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: An attribute does not contain required content. The name of the attribute and the name of the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD012S **The XML structure of the *member_name* environment index member is not valid. The content length for the *element_name* element exceeds *maximum_number* characters.**

Explanation: An element contains too many characters. The name of the element and the maximum number of allowed characters are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD013S The XML structure of the *member_name* environment index member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.

Explanation: The environment index member contains an unknown attribute. The name of the unknown attribute and the name of the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD050S The following LPAR serial number is duplicated in the environment index member: *serial_number*.

Explanation: The environment index member contains duplicate LPAR serial numbers. The duplicate serial number is indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD051S The following DB2 serial number is duplicated in the environment index member: *serial_number*.

Explanation: The environment index member contains duplicate DB2 serial numbers. The duplicate serial number is indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD052S The following DB2 group attach name is duplicated in the environment index member: *group_attach_name*.

Explanation: The environment index member contains duplicate group attach names.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD053S The reference to the following DB2 subsystem for a DB2 group attach name is duplicated in the environment index member: *subsystem_ID*.

Explanation: The environment index member contains duplicate references to a DB2 subsystem for a DB2 group attach name.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD054S The reference to the following DB2 subsystem for the *LPAR_name* LPAR is duplicated in the environment index member: *subsystem_ID*.

Explanation: The environment index member contains duplicate references to a DB2 subsystem for an LPAR. The duplicate subsystem ID is indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD055S The following DB2 group attach name was not found in the environment index member: *group_attach_name*.

Explanation: A group attach name that is referenced by a DB2 member does not exist in the environment index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD056S The following LPAR was not found in the environment index member: *LPAR_name*.

Explanation: The LPAR does not exist in the environment index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD057S The following LPAR is duplicated in the environment index member: *LPAR_name*.

Explanation: The environment index member contains duplicate LPARs. The name of the duplicate LPAR name is indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD100W The *member_name* product index member is not valid. The PL/I XML parser issued the following exception warning code: *code_number*.

Explanation: While determining if the product index member is valid, the PL/I XML parser issued the specified exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the specified exception warning code.

CCQD101S **The** *member_name* **product index member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the product index member is valid, the PL/I XML parser issued the specified exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the specified exception error code.

CCQD102S **The XML structure of the** *member_name* **product index member is not valid. The** *element_name* **element is unknown.**

Explanation: The specified product index member contains an unknown element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD103S **The XML structure of the** *member_name* **product index member is not valid. Content is not allowed for the** *element_name* **element, but content was found.**

Explanation: Content was found for an element that cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD104S **The XML structure of the** *member_name* **product index member is not valid. Content is required for the** *element_name* **element, but content was not found.**

Explanation: The specified element does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD105S **The XML structure of the** *member_name* **product index member is not valid. The content length for the** *element_name* **element exceeds** *maximum_number* **characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD106S **The XML structure of the** *member_name* **product index member is not valid. The** *element_name* **element cannot occur more than** *maximum_number* **times.**

Explanation: The specified element occurs too many times in the product index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD107S **The XML structure of the** *member_name* **product index member is not valid. The** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified element does not occur enough times in the product index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD108S **The XML structure of the** *member_name* **product index member is not valid. The** *attribute_name* **attribute in the** *element name* **element cannot occur more than** *maximum_number* **times.**

Explanation: An attribute occurs too many times. The name of the attribute and the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD109S **The XML structure of the** *member_name* **product index member is not valid. The** *attribute_name* **attribute in the** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified attribute does not occur enough times in the product index member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD110S **The XML structure of the** *member_name* **product index member is not valid. Content is not allowed for the** *attribute_name* **attribute in the** *element_name* **element, but content was found.**

Explanation: An attribute cannot contain content. The name of the attribute and the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD111S **The XML structure of the** *member_name* **product index member is not valid. Content is required for the** *attribute_name* **attribute in the** *element_name* **element, but content was not found.**

Explanation: An attribute requires content. The name of the attribute and the name of the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD112S **The XML structure of the** *member_name* **product index member is not valid. The content length for the** *element_name* **element exceeds** *maximum_number* **characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD113S **The XML structure of the** *member_name* **product index member is not valid. The** *attribute_name* **attribute in the** *element_name* **element is unknown.**

Explanation: The specified attribute in the product index member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD118S **The content of the** *member_name* **product index member is not valid. The** *configuration_ID* **configuration ID for the** *configuration-name* **configuration name is not unique.**

Explanation:

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD120S **The content of the** *member_name* **product index member is not valid. The pack ID** *pack_ID* **that is referenced by product prefix** *product_prefix* **in the metadata library** *library_name* **could not be found.**

Explanation: The specified pack ID could not be found in the metadata library.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD121I **The specified pack contains the** *component_name***, which was previously specified as a stand-alone product.**

Explanation: The specified component of the pack was previously specified as a stand-alone product.

System action: None.

User response: No action is required.

CCQD122I **The specified component metadata library was previously specified as part of the** *pack_name***.**

Explanation: The specified metadata library for the component was previously specified as part of a pack.

System action: None.

User response: No action is required.

CCQD123E **The customization library name** *library_name* **is being used by another product or component. Specify another customization library qualifier on the Tools Customizer Settings panel.**

Explanation: A different product or component is using the specified customization library.

System action: None.

User response: Specify another customization library qualifier on the Tools Customizer Settings panel.

CCQD300W **The** *member_name* **product environment member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the product environment member is valid, the PL/I XML parser issued the specified exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the specified exception warning code.

CCQD301S **The** *member_name* **product environment member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the product environment member is valid, the PL/I XML parser issued the specified exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS*

Programming Guide for more information about the specified exception error code.

CCQD302S **The XML structure of the *member_name* product environment member is not valid. The *element_name* element is unknown.**

Explanation: The specified product environment member contains an unknown element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD303S **The XML structure of the *member_name* product environment member is not valid. Content is not allowed for the *element_name* element, but content was found.**

Explanation: Content was found for an element that cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD304S **The XML structure of the *member_name* product environment member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD305S **The XML structure of the *member_name* product environment member is not valid. The content length for the *element_name* element exceeds *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD306S **The XML structure of the *member_name* product environment member is not valid. The *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified element occurs too many times in the product environment member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD307S **The XML structure of the *member_name* product environment member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times in the product environment member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD308S **The XML structure of the *member_name* product environment member is not valid. The *attribute_name* attribute in the *element name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times. The name of the attribute and the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD309S **The XML structure of the *member_name* product environment member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute does not occur enough times in the product environment member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD310S **The XML structure of the *member_name* product environment member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot contain content. The name of the attribute and the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD311S **The XML structure of the *member_name* product environment member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute requires content. The name of the attribute and the name of the element that contains it are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD312S **The XML structure of the *member_name* product environment member is not valid. The content length for the *element_name* element exceeds *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD313S **The XML structure of the *member_name* product environment member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the product environment member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD350I **The *subsystem_ID* DB2 subsystem is associated with this product.**

Explanation: The specified DB2 subsystem was added and saved in the Tools Customizer data store for the product to be customized.

System action: Processing continues.

User response: No action is required.

CCQD351I **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name is associated with this product.**

Explanation: The specified DB2 member for the group attach name was added and saved in the Tools Customizer data store for the product to be customized.

System action: Processing continues.

User response: No action is required.

CCQD352I **The *group_attach_name* DB2 group attach name is associated with this product.**

Explanation: The specified DB2 group attach name was added and saved in the Tools Customizer data store for the product to be customized.

System action: Processing continues.

User response: No action is required.

CCQD353E **The *subsystem_ID* DB2 subsystem is already associated with this product.**

Explanation: The specified DB2 subsystem cannot be added for the product to be customized because it already exists in the product environment in the data store.

System action: None.

User response: Ensure that the DB2 subsystem is specified correctly. If the problem persists, contact IBM Software Support.

CCQD354E **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name is already associated with this product.**

Explanation: The specified DB2 member for the group attach name cannot be added for the product to be customized because it already exists in the product environment in the data store.

System action: None.

User response: Ensure that the DB2 group attach name is specified correctly. If the problem persists, contact IBM Software Support.

CCQD355E **The *group_attach_name* DB2 group attach name is already associated with this product.**

Explanation: The specified DB2 group attach name cannot be added for the product to be customized because it already exists in the product environment in the data store.

System action: Processing stops.

User response: Ensure that the DB2 group attach name is specified correctly. If the problem persists, contact IBM Software Support.

CCQD356S **The *library_name* metadata library is already associated with the maximum number of allowed DB2 entries for this product.**

Explanation: The specified metadata library cannot be associated with more DB2 entries because it is already associated with the number of DB2 entries that are allowed.

System action: Processing stops.

User response: Delete an associated DB2 entry, and associate the specified library with another DB2 entry again.

CCQD357I **The *subsystem_ID* DB2 subsystem is unassociated with this product.**

Explanation: The specified DB2 SSID was unassociated with the product that you are customizing.

System action: Processing continues.

User response: No action is required.

CCQD358I **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name is unassociated with this product.**

Explanation: The specified DB2 member for the DB2 group attach name was unassociated with the product that you are customizing.

System action: Processing continues.

User response: No action is required.

CCQD359I **The *group_attach_name* DB2 group attach name is unassociated with this product.**

Explanation: The specified DB2 group attach name was unassociated with the product that you are customizing.

System action: Processing continues.

User response: No action is required.

CCQD360S **The *library_name* metadata library is not associated with the specified DB2 subsystem *subsystem_ID*.**

Explanation: The specified DB2 subsystem and metadata library are not associated with each other.

System action: None.

User response: Ensure that the DB2 subsystem and the metadata library are associated. If the problem persists, contact IBM Software Support.

CCQD361S **The *library_name* metadata library is not associated with the specified DB2 data sharing group member *member_name* for the *group_attach_name* DB2 group attach name.**

Explanation: The specified DB2 data sharing group member for the group attach name and metadata library are not associated with each other.

System action: None.

User response: Ensure that the DB2 data sharing group member for the group attach name and the metadata library are associated. If the problem persists, contact IBM Software Support.

CCQD362S **The *library_name* metadata library is not associated with the specified *group_attach_name* DB2 group attach name.**

Explanation: The specified DB2 group attach name and metadata library are not associated with each other.

System action: None.

User response: Ensure that the DB2 group attach name and the metadata library are associated. If the problem persists, contact IBM Software Support.

CCQD400W **The customization parser issued the *code_number* warning code while it parsed the product customization member *member_name*. See the PL/I programming guide for more information about this XML parser continuable exception code.**

Explanation: While determining if the specified member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the warning.

CCQD401S **The customization parser issued the *code_number* error code while it parsed the product customization member *member_name*. See the PL/I programming guide for more information about this XML parser terminating exception code.**

Explanation: While determining if the specified member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the error.

CCQD500W **The *data_set_name* data store data set was not found.**

Explanation: Tools Customizer could not find the specified data store data set.

System action: None.

User response: No action is required.

CCQD501W The *data_set_name* data store data set was not found, so it was created.

Explanation: Tools Customizer created the specified data set because it could not be found.

System action: None.

User response: No action is required.

CCQD502E The *data_set_name* data store data set is not writable.

Explanation: Tools Customizer cannot write to the specified data set.

System action: None.

User response: Ensure that the data set is writable.

CCQD503E The *data_set_name* data store data set could not be opened with the *disposition_type* disposition.

Explanation: Tools Customizer could not open the data set with the specified disposition.

System action: Processing stops.

User response: Ensure that you have WRITE authority access to this data set.

CCQD504E The *data_set_name* data store data set could not be opened with the *option_name* option.

Explanation: Tools Customizer could not open the data set with the specified option.

System action: Processing stops.

User response: Ensure that you have WRITE authority access to this data set.

CCQD505E The *data_set_name* data store data set could not be created.

Explanation: Tools Customizer could not create the specified data set.

System action: Processing stops.

User response: Ensure that you have the authority to create data sets and that the DASD is not full.

CCQD510I The DB2 SSID and DB2 group attach name were created.

Explanation: The DB2 SSID and DB2 group attach name were created and saved in the data store.

System action: None.

User response: No action is required.

CCQD511E The DB2 entry already exists in the list of DB2 entries to be associated.

Explanation: The DB2 entry cannot be added because it already exists in the list of DB2 entries to be associated.

System action: None.

User response: Specify a different DB2 entry.

CCQD512S An error occurred while a DB2 entry was being created.

Explanation: A severe error occurred while a DB2 entry was being created.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD513E The specified DB2 entry already exists and is associated with the current product on the Customizer Workplace panel.

Explanation: The DB2 entry cannot be added because it already exists, and it is already associated with the product to be customized.

System action: None.

User response: Press F3 to go to the Customizer Workplace panel to see the DB2 entry, or specify a different DB2 entry.

CCQD514E A value is required for a DB2 subsystem, a DB2 group attach name, or both before they can be created.

Explanation: Required information is missing. A DB2 subsystem, a DB2 group attach name, or both must be specified.

System action: None.

User response: Specify a DB2 subsystem, a DB2 group attach name, or both.

CCQD515E The specified DB2 entry already exists in the list of DB2 entries and is already associated with the current product.

Explanation: The DB2 entry has already been created and associated with the product that you want to customize.

System action: None.

User response: Specify a different DB2 entry.

CCQD516E The specified DB2 entry already exists in the list of DB2 entries on the Associate DB2 Entry with Product panel but is not associated with the current product.

Explanation: The DB2 entry exists, but it must be associated with the product to be customized.

System action: None.

User response: On the Customizer Workplace panel, issue the ASSOCIATE command to associate the DB2 entry with the product.

CCQD517S An error occurred while a DB2 entry was being copied.

Explanation: A severe error occurred while a DB2 entry was being copied

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD518E A value is required for a DB2 subsystem, a DB2 group attach name, or both before they can be copied.

Explanation: Required information is missing. A DB2 subsystem, a DB2 group attach name, or both must be specified.

System action: None.

User response: Specify a DB2 subsystem, a DB2 group attach name, or both.

CCQD519I The DB2 entry was copied.

Explanation: The DB2 entry was copied and saved in the Tools Customizer data store.

System action: None.

User response: No action is required.

CCQD520S The DB2 entry was copied to the list of DB2 entries but was not associated because the product is already associated with the allowed number of DB2 entries.

Explanation: The DB2 entry was not completely copied because a product can be associated with only 1200 DB2 entries.

System action: Processing stops.

User response: Remove a DB2 entry from the list, and copy the specified DB2 entry again.

CCQD521E *Line_command* **is not a valid line command.**

Explanation: The specified line command is not valid. Valid line commands are on the panel.

System action: Processing stops.

User response: Specify a valid line command.

CCQD522E The *subsystem_ID* **DB2 subsystem ID occurs more than once in the list. Each row must be unique.**

Explanation: The specified DB2 subsystem ID can be used only once.

System action: Processing stops.

User response: Specify a different DB2 subsystem ID.

CCQD523E The *group_attach_name* **DB2 group attach name occurs more than once in the list. Each row must be unique.**

Explanation: The specified DB2 group attach name can be used only once.

System action: Processing stops.

User response: Specify a different DB2 group attach name.

CCQD524E The *member_name* **DB2 member for the DB2 group attach name occurs more than once in the list. Each row must be unique.**

Explanation: The specified DB2 member for the DB2 group attach name can be used only once.

System action: Processing stops.

User response: Specify a different DB2 member for the DB2 group attach name.

CCQD525I The DB2 entries were created.

User response: No action is required.

CCQD526E The *subsystem_ID* **DB2 subsystem ID occurs more than once in the list. Each DB2 subsystem ID must be unique.**

Explanation: The specified DB2 subsystem ID can be used only once.

System action: Processing stops.

User response: Specify a different DB2 subsystem ID.

CCQD527I DB2 group attach names cannot be created during the copy process.

Explanation: The ability to create DB2 group attach names is not available during the copy process.

System action: None.

User response: Create DB2 group attach names by issuing the CREATE command on the Customizer Workplace panel.

CCQD528E The *metadata_library* metadata library is already associated with *number* DB2 entries. The maximum number of associated DB2 entries for this &CCQMPOPL is 256.

Explanation:

System action: Processing stops.

User response:

CCQD529I At least one row is required.

CCQD560E The *subsystem_ID* DB2 subsystem already exists and is associated with the current product on the Customizer Workplace panel.

Explanation: The specified DB2 subsystem exists and is associated with the product that you are customizing.

System action: None.

User response: Specify another DB2 subsystem.

CCQD561E The *member_name* DB2 member for the *group_attach_name* DB2 group attach name already exists and is associated with the current product on the Customizer Workplace panel.

Explanation: The specified DB2 data sharing group for the DB2 group attach name exists and is associated with the product that you are customizing.

System action: None.

User response: Specify another DB2 subsystem.

CCQD562E The *group_attach_name* DB2 group attach name already exists and is associated with the current product on the Customizer Workplace panel.

Explanation: The specified DB2 group attach name exists and is associated with the product that you are customizing. The subsystem is in the table on the Customizer Workplace panel.

System action: None.

User response: Specify another DB2 group attach name.

CCQD563E A value is required for a DB2 subsystem, a DB2 group attach name, or both before they can be created.

Explanation: A DB2 subsystem, a DB2 group attach name, or both are not specified so one or both of them cannot be created.

System action: None.

User response: Specify a value for the DB2 subsystem, the DB2 group attach name, or both.

CCQD565E The *subsystem_ID* DB2 subsystem already exists in the list of DB2 entries and is already associated with the current product.

Explanation: The specified subsystem is already associated.

System action: None.

User response: Specify a different DB2 subsystem.

CCQD566E The *member_name* DB2 member for the *group_attach_name* DB2 group attach name already exists in the list of DB2 entries and is already associated with the current product.

Explanation: The specified DB2member is already associated.

System action: None.

User response: Specify a different DB2 member.

CCQD567E The *group_attach_name* DB2 group attach name already exists in the list of DB2 entries and is already associated with the current product.

Explanation: The specified DB2 group attach name is already associated.

System action: None.

User response: Specify another DB2 group attach name.

CCQD568I To customize *product_name*, at least one DB2 entry must be associated with this product.

Explanation: The specified product requires at least one associated DB2 entry.

System action: None.

User response: To continue the customization process

for the specified product, associate one or more DB2 entries with it.

CCQD569I **To customize the *product_name* product configuration, at least one DB2 entry must be associated with this configuration.**

Explanation: The configuration for the specified product requires at least one associated DB2 entry.

System action: None.

User response: To continue the customization process for the configuration of the specified product, associate one or more DB2 entries with the configuration.

CCQD577W **The *mode_name* DB2 mode of the *subsystem_ID* DB2 subsystem is not supported by the product.**

Explanation: The product does not support the specified DB2 mode.

System action: None.

User response: Specify a supported DB2 mode.

CCQD578W **The *mode_name* DB2 mode of the *member_name* DB2 member for the DB2 group is not supported by the product.**

Explanation: The product does not support the specified DB2 mode.

System action: None.

User response: Specify a supported DB2 mode.

CCQD579W **The *mode_name* DB2 mode of the *group_name* DB2 group attach name is not supported by the product.**

Explanation: The product does not support the specified DB2 mode.

System action: None.

User response: Specify a supported DB2 mode.

CCQD580S **The *subsystem_ID* DB2 subsystem was copied to the list of DB2 entries but was not associated because the product is already associated with the allowed number of DB2 entries.**

Explanation: The copied DB2 subsystem was not associated with the product because the product is associated with the maximum number of DB2 entries.

System action: None.

User response: Remove an associated DB2 entry and associate the product with the copied DB2 subsystem.

CCQD581S **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name was copied to the list of DB2 entries but was not associated because the product is already associated with the allowed number of DB2 entries.**

Explanation: The copied DB2 member for the DB2 group attach name was not associated with the product because the product is associated with the maximum number of DB2 entries.

System action: None.

User response: Remove an associated DB2 entry and associate the product with the copied DB2 member.

CCQD582S **The *group_attach_name* DB2 group attach name was copied to the list of DB2 entries but was not associated because the product is already associated with the allowed number of DB2 entries.**

Explanation: The copied DB2 group attach name was not associated with the product because the product is associated with the maximum number of DB2 entries.

System action: None.

User response: Remove an associated DB2 entry and associate the product with the copied DB2 group attach name.

CCQD584I **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name is copied to the *subsystem_ID* DB2 subsystem.**

Explanation: The specified DB2 member was copied.

System action: None.

User response: No action is required.

CCQD585I **The *group_attach_name* DB2 group attach name cannot be copied because a DB2 member is required.**

Explanation: The specified DB2 group attach name was not copied because a DB2 member was missing.

System action: None.

User response: No action is required.

CCQD586S **The current LPAR is *LPAR_name*, but the data store contains information about the *LPAR_name* LPAR. You must use the *LPAR_name* LPAR to customize the product.**

Explanation: The LPAR that is stored in the data store data set must be used to customize the product.

System action: Processing stops.

User response: Use the LPAR that is stored in the data store data set.

CCQD587W **The** *level_number* **DB2 level of the** *subsystem_name* **DB2 subsystem is not supported by the product.**

Explanation: The product does not support the specified DB2 level.

System action: Processing continues.

User response: Specify a supported level of DB2.

CCQD588W **The** *level_number* **DB2 level of the** *member_name* **DB2 member of the** *group_name* **DB2 group is not supported by the product.**

Explanation: The product does not support the specified DB2 level.

System action: Processing continues.

User response: Specify a supported level of DB2.

CCQD589W **The** *level_number* **DB2 level of the** *group_name* **DB2 group attach name is not supported by the product.**

Explanation: The product does not support the specified DB2 level.

System action: Processing continues.

User response: Specify a supported level of DB2.

CCQD593I **The** *subsystem_ID* **DB2 subsystem was deleted.**

User response: No action is required.

CCQD594I **The** *member_name* **DB2 for the** *group_attach_name* **DB2 group attach name was deleted.**

User response: No action is required.

CCQD595I **The** *group_attach_name* **DB2 group attach name was deleted.**

User response: No action is required.

CCQD596E **The** *subsystem_ID* **DB2 subsystem was not deleted.**

Explanation: An internal error occurred while the specified DB2 subsystem was being deleted.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD597E **The** *member_name* **DB2 member for the** *group_attach_name* **DB2 group attach name was not deleted.**

Explanation: An internal error occurred while the specified DB2 member was being deleted.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD598E **The** *group_attach_name* **DB2 group attach name was not deleted.**

Explanation: An internal error occurred while the specified DB2 group attach name was being deleted.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD600W **The** *member_name* **product customization member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the XML structure of the product customization member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQD601S **The** *member_name* **product customization member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the XML structure of the product customization member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception error code.

CCQD602S **The XML structure of the** *member_name* **product customization member is not valid. The** *element_name* **element is unknown.**

Explanation: The data store member contains an unknown element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD603S **The XML structure of the *member_name* product customization member is not valid. Content is not allowed for the *element_name* element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD604S **The XML structure of the *member_name* product customization member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD605S **The XML structure of the *member_name* product customization member is not valid. The content length for the *element_name* element exceeds *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD606S **The XML structure of the *member_name* product customization member is not valid. The *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified element occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD607S **The XML structure of the *member_name* product customization member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD608S **The XML structure of the *member_name* product customization member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD609S **The XML structure of the *member_name* product customization member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD610S **The XML structure of the *member_name* product customization member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD611S **The XML structure of the *member_name* product customization member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD612S **The XML structure of the *member_name* product customization member is not valid. The content length for the *element_name* element exceeds *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD613S **The XML structure of the** *member_name* **product customization member is not valid. The** *attribute_name* **attribute in the** *element_name* **element is unknown.**

Explanation: The specified attribute in the data store member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD614S **The content of the** *member_name* **product customization member is not valid. The value of the** *element_name* **element is not valid. The value is** *value_name***.**

Explanation: The specified value is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQD700W **The** *member_name* **DB2 data member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the XML structure of the DB2 data member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQD701S **The** *member_name* **DB2 data member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the XML structure of the DB2 data member is valid, the PL/I XML parser issued an exception error code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception error code.

CCQD750W **The** *value_number* **value in the DB2 parameter** *parameter_name* **was skipped because only** *maximum_number* **values are allowed.**

Explanation: The specified value was skipped because it exceeds the number of allowed values in the DB2 parameter.

System action: Processing continues.

User response: No action is required. To stop this message from being issued, remove the extra values from the DB2 parameter.

CCQD800W **The** *member_name* **LPAR data member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the XML structure of the LPAR data member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQD801S **The** *member_name* **LPAR data member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the XML structure of the LPAR data member is valid, the PL/I XML parser issued an exception error code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception error code.

CCQD850W **The** *value_number* **value in the LPAR parameter** *parameter_name* **was skipped because only** *maximum_number* **values are allowed.**

Explanation: The specified value was skipped because it exceeds the number of allowed values in the LPAR parameter.

System action: Processing continues.

User response: No action is required. To stop this message from being issued, remove the extra values from the LPAR parameter.

CCQD851I **The** *subsystem_ID* **DB2 subsystem is copied to the** *member_name* **DB2 member for the** *group_attach_name* **DB2 group attach name.**

User response: No action is required.

CCQD852I **The** *member_name* **DB2 member for the** *group_attach_name* **DB2 group attach name is copied to the** *member_name* **DB2 member for the** *group_attach_name* **DB2 group attach name.**

User response: No action is required.

CCQD854I **The *member_name* DB2 member for the *group_attach_name* DB2 group 'attach name is copied to multiple DB2 entries.**

User response: No action is required.

CCQD900W **The *member_name* product data member is not valid. The PL/I XML parser issued the following exception warning code: *code_number*.**

Explanation: While determining if the XML structure of the product data member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQD901S **The *member_name* product data member is not valid. The PL/I XML parser issued the following exception error code: *code_number*.**

Explanation: While determining if the XML structure of the product data member is valid, the PL/I XML parser issued an exception error code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQD950W **The *value_number* value in the product parameter *parameter_name* was skipped because only *maximum_number* values are allowed.**

Explanation: The specified value was skipped because it exceeds the number of allowed values in the product parameter.

System action: Processing continues.

User response: No action is required. To stop this message from being issued, remove the extra values from the product parameter.

CCQD960I **The *subsystem_ID* DB2 subsystem was changed to the *member_name* DB2 member for the *group_attach_name* DB2 group attach name.**

User response: No action is required.

CCQD961I **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name was changed to the *subsystem_ID* DB2 subsystem.**

User response: No action is required.

CCQD962I **The *member_name* DB2 member for the *group_attach_name* DB2 group attach name was changed to the *member_name* DB2 member for the *group_attach_name* DB2 group attach name.**

User response: No action is required.

CCQD963E **The DB2 group attach name cannot be blank when the DB2 subsystem ID is blank.**

Explanation: A DB2 group attach name, DB2 subsystem ID, or both must be specified.

System action: Processing stops.

User response: Specify a DB2 group attach name, DB2 subsystem ID, or both.

CCQE000S **The specified message field name or message *message_ID* was not found.**

Explanation: An error occurred while displaying a message field name or the specified message.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQE001E **An incorrect trace level was specified. Valid trace levels are 0 - 4.**

Explanation: A wrong trace level was specified. Valid trace levels are 0 - 4.

System action: Processing stops.

User response: Specify a valid trace level 0 - 4.

CCQH001W **The specified option *option_name* is not valid.**

Explanation: The option that was specified is not a valid option on the panel.

System action: Tools Customizer stops.

User response: Specify a valid option on the panel.

CCQH006W **Before you customize a product, verify your user settings.**

Explanation: The user settings must be verified before a product can be customized.

System action: Tools Customizer stops.

User response: Verify the user settings.

CCQH007E Check the user settings. One or more current values are not valid.

Explanation: One or more of the values in the user settings is not valid.

System action: Tools Customizer stops.

User response: Ensure that the specified values for the user settings are valid.

CCQH008W Before you use Tools Customizer, you must select option 0 to verify your user settings.

Explanation: The user settings must be changed before a product can be customized.

System action: Tools Customizer stops.

User response: Change the user settings.

CCQH009E You must select option 0 to change your user settings.

Explanation: User settings must be changed before a product can be customized.

System action: Tools Customizer stops.

User response: Change the user settings.

CCQI000W The XML structure of the *member_name* **DB2 parameter metadata member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the DB2 parameter metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI001S The XML structure of the *member_name* **DB2 parameter metadata member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the DB2 parameter metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI002S The XML structure of the *member_name* **DB2 parameter metadata member is not valid. The** *element name* **element is unknown.**

Explanation: The specified element in the DB2 parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI003S The XML structure of the *member_name* **DB2 parameter metadata member is not valid. Content is not allowed for the** *element_name* **element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI004S The XML structure of the *member_name* **DB2 parameter metadata member is not valid. Content is required for the** *element_name* **element, but content was not found.**

Explanation: The specified element requires content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI005S The XML structure of the *member_name* **DB2 parameter metadata member is not valid. The content length for the** *element_name* **element cannot exceed** *maximum_number* **characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI006S The XML structure of the *member_name* **DB2 parameter metadata member is not valid. The content length for the** *element_name* **element must be at least** *minimum_number* **characters.**

Explanation: The specified element does not contain enough characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI007S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI008S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI009S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute did not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI010S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot have content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI011S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI012S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI013S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the DB2 parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI014S **The content of the *member_name* DB2 parameter metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value of the element is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI015S **The content of the DB2 parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified value of the attribute is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI016S **The content of the DB2 parameter metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type is not a valid data type.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI017S **The content of the DB2 parameter metadata member is not valid because the data type of the** *attribute_name* **attribute in the** *element_name* **element is incorrect. The value of the attribute is** *value_name***.**

Explanation: The specified data type is not a valid data type.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI050S **The** *member_name* **DB2 parameter metadata member was not found in the** *data_set_name* **data set.**

Explanation: Tools Customizer could not find the specified DB2 parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI051S **The** *parameter_name* **LPAR parameter in the** *template_name* **template does not have associated metadata in the** *member_name* **LPAR parameter metadata member.**

Explanation: The specified template does not contain metadata for an LPAR parameter. The name of the LPAR parameter metadata member, the name of the LPAR parameter, and the name of the template are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI052S **The** *parameter_name* **product parameter in the** *template_name* **template does not have associated metadata in the** *member_name* **product parameter metadata member.**

Explanation: The specified template does not contain metadata for a product parameter. The name of the product parameter metadata member, the name of the product parameter, and the name of the template are indicated in the message text.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI053E **The following metadata data set was not found:** *data_set_name***.**

Explanation: Tools Customizer could not find the specified metadata data set.

System action: Processing stops.

User response: Ensure that the metadata data set is specified correctly. If the problem persists, contact IBM Software Support.

CCQI054E **The following metadata data set could not be opened:** *data_set_name***.**

Explanation: Tools Customizer could not open the specified LPAR metadata data set.

System action: Processing stops.

User response: Ensure the metadata data set was specified correctly.

CCQI055S **The CCQ$$DB2 DB2 parameter metadata member was not found in the** *data_set_name* **Tools Customizer metadata data set.**

Explanation: Tools Customizer could not find the DB2 parameter metadata member in the specified Tools Customizer metadata data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI056S **The CCQ$$LPR LPAR parameter metadata member was not found in the** *data_set_name* **data set.**

Explanation: Tools Customizer could not find the specified LPAR parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI057S **The** *member_name* **product parameter metadata member was not found in the** *data_set_name* **data set.**

Explanation: The product parameter metadata member was not found in the specified data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI058I *Product_name* **does not have any DB2 parameters.**

Explanation: DB2 parameters are not required to customize the specified product.

System action: Processing continues.

User response: No action is required.

CCQI059I *Product_name* **does not have any LPAR parameters.**

Explanation: LPAR parameters are not required to customize the specified product.

System action: Processing continues.

User response: No action is required.

CCQI060S **The *parameter_name* DB2 parameter in the *task_description* task condition does not have associated metadata in the *member_name* DB2 parameter metadata member.**

Explanation: Associated metadata is missing for the specified DB2 parameter in a task.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI061S **The *parameter_name* LPAR parameter in the *task_description* task condition does not have associated metadata in the *member_name* LPAR parameter metadata member.**

Explanation: Associated metadata is missing for the specified LPAR parameter in a task.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI062S **The *parameter_name* product parameter in the *task_description* task condition does not have associated metadata in the *member_name* product parameter metadata member.**

Explanation: Associated metadata is missing for the specified product parameter in a task.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI063S **The *parameter_name* DB2 parameter in the *task_description* task and the *step_description* step does not have associated metadata in the *member_name* DB2 parameter metadata member.**

Explanation: Associated metadata is missing for the specified DB2 parameter in a task and step.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI064S **The *parameter_name* LPAR parameter in the *task_description* task and the *step_description* step does not have associated metadata in the *member_name* LPAR parameter metadata member.**

Explanation: Associated metadata is missing for the specified LPAR parameter in a task and step.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI065S **The *parameter_name* product parameter in the *task_description* task and the *step_description* step does not have associated metadata in the *member_name* parameter metadata member.**

Explanation: Associated metadata is missing for the specified parameter in a task and step.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI066S **The *parameter_name* DB2 parameter in the *task_description* task, *step_description* step, and *template_name* template condition does not have associated metadata in the *member_name* DB2 parameter metadata member.**

Explanation: Associated metadata is missing for the specified DB2 parameter in a task, step, and template.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI067S **The *parameter_name* LPAR parameter in the *task_description* task, *step_description* step, and *template_name* template condition does not have associated metadata in the *member_name* LPAR parameter metadata member.**

Explanation: Associated metadata is missing for the specified LPAR parameter in a task, step, and template.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI068S **The *parameter_name* product parameter in the *task_description* task, *step_description* step, and *template_name* template condition does not have associated metadata in the *member_name* product parameter metadata member.**

Explanation: Associated metadata is missing for the specified product parameter in a task, step, and template.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI069S **Product metadata does not support multiple configurations, but the *template_name* product template contains the *parameter_name* parameter. Enable multiple configurations support for this product, and try again.**

Explanation: The specified template contains a parameter for multiple configurations, but the product is not enabled to support multiple configurations.

System action: Processing stops.

User response: Enable multiple configurations support, and try again.

CCQI070E **The** *parameter_name* **DB2 parameter metadata member is not valid. The default length for the** *parameter-element_name* **parameter element exceeds the length of the parameter. The default length is** *default_length*, **and the specified length is** *specified_length*. **The default length will be truncated accordingly.**

Explanation: The specified length cannot be shorter than the default length.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI071E **The** *parameter_name* **LPAR parameter metadata member is not valid. The default length for the** *parameter-element_name* **parameter element exceeds the length of the parameter. The default length is** *default_length*, **and the specified length is** *specified_length*. **The default length will be truncated accordingly.**

Explanation: The specified length cannot be shorter than the default length.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI072E **The** *parameter_name* **product parameter metadata member is not valid. The default length for the** *parameter-element_name* **parameter element exceeds the length of the parameter. The default length is** *default_length*, **and the specified length is** *specified_length*. **The default length will be truncated accordingly.**

Explanation: The specified length cannot be shorter than the default length.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI073S **The XML structure of the** *member_name* **DB2 parameter metadata member is not valid. The following value of the** *attribute_name* **attribute in the** *element_name* **element already exists:** *value_name*.

Explanation: The specified value already exists for an attribute.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI074S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The following value of the** *attribute_name* **attribute in the** *element_name* **element already exists:** *value_name*.

Explanation: The specified value already exists for an attribute.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI075S **The XML structure of the** *member_name* **product parameter metadata member is not valid. The following value of the** *attribute_name* **attribute in the** *element_name* **element already exists:** *value_name*.

Explanation: The specified value already exists for an attribute.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI076S **The XML structure of the** *member_name* **DB2 parameter metadata member is not valid. The** *parameter_name* **parameter refers to the** *section-name* **section. This section was not found in the DB2 parameter metadata member.**

Explanation: The specified value already exists for an attribute.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI077S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The** *parameter_name* **parameter refers to the** *section-name* **section. This section was not found in the LPAR parameter metadata member.**

Explanation: The specified parameter refers to a

section that is not in the LPAR parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI078S **The XML structure of the *member_name* product parameter metadata member is not valid. The *parameter_name* parameter refers to the *section-name* section. This section was not found in the product parameter metadata member.**

Explanation: The specified parameter refers to a section that is not in the product parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI080S **The content of the *member_name* DB2 parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified value for an attribute in the DB2 parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI081S **The content of the *member_name* LPAR parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified value for an attribute in the LPAR parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI082S **The content of the *member_name* product parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified value for an attribute in the product parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI090S **The product-defined DB2 parameter *parameter_name* in the *member_name* parameter metadata member references the *section_ID* section ID, but this ID does not exist in either the parameter metadata member or the DB2 parameter metadata member.**

Explanation: A section that does not exist in the parameter metadata member or the DB2 parameter metadata member is referenced by the specified DB2 parameter.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI091S **The product-defined LPAR parameter in the *member_name* parameter metadata member references the *section_ID* section ID, but this ID does not exist in either the parameter metadata member or the LPAR parameter metadata member.**

Explanation: A section that does not exist in the parameter metadata member or the LPAR parameter metadata member is being referenced by the specified LPAR parameter.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI092S **The overridden DB2 parameter *parameter_name* in the *member_name* parameter metadata member does not exist in the DB2 parameter metadata member.**

Explanation: The specified parameter does not exist.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI093S **The overridden LPAR parameter *parameter_name* in the *member_name* parameter metadata member does not exist in the LPAR parameter metadata member.**

Explanation: The specified parameter does not exist.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI094S **The CCQ$$PRD product customization parameter metadata member was not found in the *data_set_name* data set.**

Explanation: The specified data set must contain the CCQ$$PRD product customization parameter metadata member

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI100W **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the LPAR parameter metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI101S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the LPAR parameter metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI102S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The** *element_name* **element is unknown.**

Explanation: The specified element in the LPAR parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI103S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. Content is not allowed for the** *element_name* **element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI104S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. Content is required for the** *element_name* **element, but content was not found.**

Explanation: The specified element requires content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI105S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The content length for the** *element_name* **element cannot exceed** *maximum_number* **characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI106S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The content length for the** *element_name* **element must be at least** *minimum_number* **characters.**

Explanation: The specified element does not contain enough characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI107S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI108S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The** *attribute_name* **attribute in the** *element_name* **element cannot occur more than** *maximum_number* **times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI109S **The XML structure of the *member_name* LPAR parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute did not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI110S **The XML structure of the *member_name* LPAR parameter metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot have content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI111S **The XML structure of the *member_name* LPAR parameter metadata member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI112S **The XML structure of the *member_name* LPAR parameter metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI113S **The XML structure of the *member_name* LPAR parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the LPAR parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI114S **The content of the *member_name* LPAR parameter metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an element in the LPAR parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI115S **The content of the *member_name* LPAR parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified value for an attribute in the LPAR parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI116S **The content of the *member_name* LPAR parameter metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an element in the LPAR parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI117S **The content of the *member_name* LPAR parameter metadata member is not valid because the data type of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an attribute in the LPAR parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI120S **The XML structure of the *member_name* DB2 parameter metadata member is not valid. The *element_name* element in the *parameter_name* parameter contains duplicate values for the *element_name* element. The duplicate value is *value_name*.**

Explanation: An element contains the specified duplicate value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI121S **The XML structure of the** *member_name* **LPAR parameter metadata member is not valid. The** *element_name* **element in the** *parameter_name* **parameter contains duplicate values for the** *element_name* **element. The duplicate value is** *value_name***.**

Explanation: An element contains the specified duplicate value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI122S **The XML structure of the** *member_name* **parameter metadata member is not valid. The** *element_name* **element in the** *parameter_name* **parameter contains duplicate values for the** *element_name* **element. The duplicate value is** *value_name***.**

Explanation: An element contains the specified duplicate value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI123S **The XML structure of the** *member_name* **discover metadata member is not valid. The** *element_name* **element in the** *parameter_name* **parameter contains duplicate values for the** *element_name* **element. The duplicate value is** *value_name***.**

Explanation: An element contains the specified duplicate value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI124S **The XML structure of the** *member_name* **product customization parameter metadata member is not valid. The** *element_name* **element in the** *parameter_name* **parameter contains duplicate values for the** *element_name* **element. The duplicate value is** *value_name***.**

Explanation: An element contains the specified duplicate value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI200W **The XML structure of the** *member_name* **information metadata member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the information metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI201S **The XML structure of the** *member_name* **information metadata member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the information metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI202S **The XML structure of the** *member_name* **information metadata member is not valid. The** *element name* **element is unknown.**

Explanation: The specified element in the information metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI203S **The XML structure of the** *member_name* **information metadata member is not valid. Content is not allowed for the** *element_name* **element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI204S **The XML structure of the *member_name* information metadata member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element requires content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI205S **The XML structure of the *member_name* information metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI206S **The XML structure of the *member_name* information metadata member is not valid. The content length for the *element_name* element must be at least *minimum_number* characters.**

Explanation: The specified element does not contain enough characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI207S **The XML structure of the *member_name* information metadata member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI208S **The XML structure of the *member_name* information metadata member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI209S **The XML structure of the *member_name* information metadata member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute did not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI210S **The XML structure of the *member_name* information metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot have content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI211S **The XML structure of the *member_name* information metadata member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI212S **The XML structure of the *member_name* information metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI213S **The XML structure of the *member_name* information metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the information metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI214S **The content of the *member_name* information metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an element in the information metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI215S **The content of the *member_name* information metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an attribute in the information metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI216S **The content of the *member_name* information metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an element in the information metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI217S **The content of the *member_name* information metadata member is not valid because the data type of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an attribute in the information metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI218S **The content of the *member_name* information metadata member is not valid. The length of the *value_name* value that of the *attribute_name* attribute is longer than the *value_name* value of the *attribute_name* attribute.**

Explanation: The first specified value cannot be longer than the second specified value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI219S **The content of the *member_name* information metadata member is not valid. The *value_name* value of the *attribute_name* attribute contains the *value_name* value.**

Explanation: The first specified value cannot be longer than the second specified value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI220S **The XML structure of the *member_name* information metadata member is not valid. Content for the *attribute_name* attribute in the *element_name* element exceed *maximum_number* characters.**

Explanation: The specified attribute contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI223S **The XML structure of the *member_name* information metadata member is not valid. The value that is specified for the DB2 Level already exists. The value is *value_name*.**

Explanation: The specified value already exists.

System action: Processing stops.

User response: Specify a different DB2 level. If the problem persists, contact IBM Software Support.

CCQI224S **The XML structure of the *member_name* information metadata member is not valid. The value that is specified for the DB2 Mode already exists. The value is *value_name*.**

Explanation: The specified value already exists.

System action: Processing stops.

User response: Specify a different DB2 mode. If the problem persists, contact IBM Software Support.

CCQI250S **The information metadata member was not found in the *data_set_name* data set.**

Explanation: Tools Customizer could not find the information metadata member in the specified data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI251E **The *member_name* member was not accessible in the *data_set_name* data set.**

Explanation: The specified member could not be accessed in the data set.

System action: Processing stops.

User response: Specify the correct metadata library.

CCQI252S **The information metadata member was not found in the *library_name* component metadata library that is part of the *library_name* pack metadata library. The name of the pack is *pack_name*.**

Explanation: The specified component metadata library does not contain the information metadata member.

System action: Processing stops.

User response: Specify the correct metadata library.

CCQI253E **The *library_name* Tools Customizer metadata library is not current. Update the metadata library on the Tools Customizer Settings panel.**

Explanation: The specified metadata library is not current.

System action: Processing stops.

User response: Specify a current metadata library on the Tools Customizer Settings panel.

CCQI300W **The XML structure of the *member_name* sequence metadata member is not valid. The PL/I XML parser issued the following exception warning code: *code_number*.**

Explanation: While determining if the sequence metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI301S **The XML structure of the *member_name* sequence metadata member is not valid. The PL/I XML parser issued the following exception error code: *code_number*.**

Explanation: While determining if the sequence metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception error code, and contact IBM Software Support.

CCQI302S **The XML structure of the *member_name* sequence metadata member is not valid. The *element_name* element is unknown.**

Explanation: The specified element in the sequence metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI303S **The XML structure of the *member_name* sequence metadata member is not valid. Content is not allowed for the *element_name* element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI304S **The XML structure of the *member_name* sequence metadata member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI305S **The XML structure of the *member_name* sequence metadata member is not valid. Content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI306S **The XML structure of the *member_name* sequence metadata member is not valid. The *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified element occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI307S **The XML structure of the *member_name* sequence metadata member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI308S **The XML structure of the *member_name* sequence metadata member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI309S **The XML structure of the *member_name* sequence metadata member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI310S **The XML structure of the *member_name* sequence metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI311S **The XML structure of the *member_name* sequence metadata member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI312S **The XML structure of the *member_name* sequence metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI313S **The XML structure of the *member_name* sequence metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the sequence metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI314S **The content of the *member_name* sequence metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an element in the sequence metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI315S **The content of the *member_name* sequence metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an attribute in the sequence metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI316S **The content of the *member_name* sequence metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an element in the sequence metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI317S **The content of the *member_name* sequence metadata member is not valid because the data type of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an attribute in the sequence metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI350S **The XML structure of the *member_name* sequence metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: A specified value for an attribute in the sequence metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI351S **The *member_name* sequence metadata member was not found in the *data_set_name* metadata data set.**

Explanation: Tools Customizer could not find the specified sequence metadata member in the metadata data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI352S **The *template_name* product template was not found in the *data_set_name* metadata data set.**

Explanation: Tools Customizer could not find the specified product template in the data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI353S **The sequence metadata member was not found in the *data_set_name* component data set that is part of the *data_set_name* pack.**

Explanation: Tools Customizer could not find the sequence metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI360S **The XML structure of the *member_name* sequence metadata member is not valid. The value of the *attribute_name* attribute in the *element_name* element already exists.**

Explanation: The specified attribute contains a value that already exists.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI361S **The XML structure of the *member_name* sequence metadata member is not valid. The condition element on the *level_type* level already contains a relational operator.**

Explanation: A relational operator already exists for the condition element on the specified level.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI362S **The XML structure of the *member_name* sequence metadata member is not valid. The condition element on the *level_type* level must contain only one content string or content number element.**

Explanation: Only one content string element or content number element can be contained in the condition element on the specified level.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI363S **The XML structure of the *member_name* sequence metadata member is not valid. The condition element in the *element_name* element with the *attribute_name* attribute must contain either the content string element or content number element.**

Explanation: Either the content string element or the content number element must be in the condition element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI400W **The XML structure of the *member_name* parameter metadata member is not valid. The PL/I XML parser issued the following exception warning code: *code_number*.**

Explanation: While determining the parameter

metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI401S **The XML structure of the** *member_name* **parameter metadata member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the parameter metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQI402S **The XML structure of the** *member_name* **parameter metadata member is not valid. The** *element name* **element is unknown.**

Explanation: The specified element in the parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI403S **The XML structure of the** *member_name* **parameter metadata member is not valid. Content is not allowed for the** *element_name* **element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI404S **The XML structure of the** *member_name* **parameter metadata member is not valid. Content is required for the** *element_name* **element, but content was not found.**

Explanation: The specified element requires content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI405S **The XML structure of the** *member_name* **parameter metadata member is not valid. The content length for the** *element_name* **element cannot exceed** *maximum_number* **characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI406S **The XML structure of the** *member_name* **parameter metadata member is not valid. The content length for the** *element_name* **element must be at least** *minimum_number* **characters.**

Explanation: The specified element does not contain enough characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI407S **The XML structure of the** *member_name* **parameter metadata member is not valid. The** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI408S **The XML structure of the** *member_name* **parameter metadata member is not valid. The** *attribute_name* **attribute in the** *element_name* **element cannot occur more than** *maximum_number* **times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI409S **The XML structure of the** *member_name* **parameter metadata member is not valid. The** *attribute_name* **attribute in the** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified attribute does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI410S **The XML structure of the *member_name* parameter metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot have content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI411S **The XML structure of the *member_name* parameter metadata member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI412S **The XML structure of the *member_name* parameter metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI413S **The XML structure of the *member_name* parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI414S **The content of the *member_name* parameter metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an element in the parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI415S **The content of the *member_name* parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an attribute in the parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI416S **The content of the *member_name* parameter metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an element in the parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI417S **The content of the *member_name* parameter metadata member is not valid because the data type of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an attribute in the parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI420S **The XML structure of the *member_name* parameter metadata member is not valid. The *element_name* element is unknown for the overridden DB2 parameter.**

Explanation:

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI421S **The XML structure of the *member_name* parameter metadata member is not valid. The *element_name* element is unknown for the overridden LPAR parameter.**

Explanation:

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI422S **The XML structure of the *member_name* parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown for the overridden DB2 parameter.**

Explanation:

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI423S **The XML structure of the *member_name* parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown for the overridden LPAR parameter.**

Explanation:

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI450S **The *member_name* product parameter metadata member was not found in the *data_set_name* data set.**

Explanation: Tools Customizer could not find the specified product parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI510W **The *data_set_name* data store data set does not exist.**

Explanation: The specified data store data set does not exist.

System action: Processing continues.

User response: Ensure that the data store data set exists.

CCQI511S **The *data_set_name* data store data set cannot be opened by using the *disposition_type* disposition.**

Explanation: The specified data store data set could not be opened with the specified disposition.

System action: Processing continues.

User response: Contact IBM Software Support.

CCQI512S **The *data_set_name* data store data set cannot be opened by using the *option-type* option.**

Explanation: The specified data store data set was unable to be opened with the specified option.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI600W **The XML structure of the *member_name* product customization parameter metadata member is not valid. The PL/I XML parser issued the following exception warning code: *code_number*.**

Explanation: While determining if the product customization parameter metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the warning.

CCQI601S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The PL/I XML parser issued the following exception error code: *code_number*.**

Explanation: While determining if the product customization parameter metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the warning.

CCQI602S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *element_name* element is unknown.**

Explanation: The specified product customization parameter metadata member contains an unknown element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI603S **The XML structure of the *member_name* product customization parameter metadata member is not valid. Content is not allowed for the *element_name* element, but content was found.**

Explanation: Content was found in an element that cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI604S **The XML structure of the *member_name* product customization parameter metadata member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI605S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The content length for the *element_name* element 'cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI606S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified element occurs too many times in the product customization parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI607S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times in the product customization parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI608S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times in the product customization parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI609S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute does not occur enough times in the product customization parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI610S **The XML structure of the *member_name* product customization parameter metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: Content was found in an element that cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI611S **The XML structure of the *member_name* product customization parameter metadata member is not valid. Content is required for the *attribute_name* attribute 'in the *element_name* element, but content was not found.**

Explanation: The specified attribute does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI612S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The content length for the *attribute_name* attribute in the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified attribute contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI613S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified product customization parameter metadata member contains an unknown attribute.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI614S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The value of the *element_name* element is not valid. The value *value_name*.**

Explanation: The specified value of the element is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI615S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The value of the *attribute_name* attribute for the *element_name* element is not valid. The value is *value_name*.**

Explanation: The specified value of the attribute is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI616S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The data type of the *element_name* element is 'not valid. The value of the element is *value_name*.**

Explanation: The specified data type is not a valid data type.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI617S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The data type of the *attribute_name* attribute for the *element_name* element is not valid. The value of the attribute is *value_name*..**

Explanation: The specified data type is not a valid data type.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI650S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The following value of the *attribute_name* attribute in the *element_name* element already exists: *value_name*.**

Explanation: The specified value for an attribute already exists.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI651S **The XML structure of the *member_name* product customization parameter metadata member is not valid. The *parameter_name* parameter refers to the following section, which was not found in the *member_name* product customization parameter metadata member: *section-name*.**

Explanation: The specified section is not in the product customization parameter metadata member.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI652S **The *member_name* product customization metadata member not valid. The default length for the *element_name* parameter element exceeds the length of the parameter. The default length is *default_length*, and the specified length is *specified_length*. The default length will be truncated accordingly.**

Explanation: The specified length cannot be shorter than the default length.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI653S **The content of the *member_name* product customization parameter metadata member is not valid. The value of the *attribute_name* attribute in the *element_name* element is not valid. The value of the attribute is *value_name*.**

Explanation: The specified value of the attribute is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI700W **The XML structure of the *member_name* solution pack metadata member is not valid. The PL/I XML parser issued the following exception warning code: *code_number*.**

Explanation: While determining if the specified solution pack metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the warning.

CCQI701S **The XML structure of the *member_name* solution pack metadata member is not valid. The PL/I XML parser issued the following exception error code: *code_number*.**

Explanation: While determining if the specified solution pack metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the error.

CCQI702S **The XML structure of the *member_name* solution pack metadata member is not valid. The *element_name* element is unknown.**

Explanation: The specified solution pack metadata member contains an unknown element.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI703S **The XML structure of the *member_name* solution pack metadata member is not valid. Content is not allowed for the *element_name* element, but content was found**

Explanation: Content was found in an element that cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI704S **The XML structure of the *member_name* solution pack metadata member is not valid. Content is required for the *element_name* element, but content was not found.**

Explanation: The specified element does not contain required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI705S **The XML structure of the *member_name* solution pack metadata member is not valid. The content length for the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI706S **The XML structure of the *member_name* solution pack metadata member is not valid. The *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified element occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI707S **The XML structure of the *member_name* solution pack metadata member is not valid. The *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI708S **The XML structure of the *member_name* solution pack metadata member is not valid. The *attribute_name* attribute in the *element_name* element cannot occur more than *maximum_number* times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI709S **The XML structure of the *member_name* solution pack metadata member is not valid. The *attribute_name* attribute in the *element_name* element must occur at least *minimum_number* times.**

Explanation: The specified attribute does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI710S **The XML structure of the *member_name* solution pack metadata member is not valid. Content is not allowed for the *attribute_name* attribute in the *element_name* element, but content was found.**

Explanation: The specified attribute cannot have content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI711S **The XML structure of the *member_name* solution pack metadata member is not valid. Content is required for the *attribute_name* attribute in the *element_name* element, but content was not found.**

Explanation: The specified attribute is missing content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI712S **The XML structure of the *member_name* solution pack metadata member is not valid. The content length for the *attribute_name* attribute in the *element_name* element cannot exceed *maximum_number* characters.**

Explanation: The specified attribute contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI713S **The XML structure of the *member_name* solution pack metadata member is not valid. The *attribute_name* attribute in the *element_name* element is unknown.**

Explanation: The specified attribute in the solution pack metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI714S **The XML structure of the *member_name* solution pack metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value of the element is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI715S **The XML structure of the *member_name* solution pack metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified value of the attribute is not a valid value.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI716S **The XML structure of the *member_name* solution pack metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type is not a valid data type.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI717S **The XML structure of the *member_name* solution pack metadata member is not valid because the data type of the *attribute_name* attribute in the *element_name* element is incorrect. The value of the attribute is *value_name*.**

Explanation: The specified data type is not a valid data type.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI720S **The XML structure of the *member_name* solution pack metadata member is not valid. The msg element is required for the *component_name* component that is not customizable.**

Explanation: The msg element is required for the specified component, which cannot be customized by using Tools Customizer.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI750S **The solution pack metadata member was not found in the** *library_name* **metadata library.**

Explanation: Tools Customizer could not find the solution pack metadata member in the specified library.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI751S **The version in the** *library_name* **solution pack metadata library is different than the version in the** *library_name* **component metadata library. The name of the pack is** *pack_name***, and the name of the component is** *component_name***.**

Explanation: The version in the solution pack metadata library does not match the version in the component metadata library.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI752S **The release in the** *library_name* **solution pack metadata library is different than the release in the** *library_name* **component metadata library. The name of the pack is** *pack_name***, and the name of the component is** *component_name***.**

Explanation: The release in the solution pack metadata library does not match the release in the component metadata library.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQI753S **The modification level in the** *library_name* **solution pack metadata library is different than the modification level in the** *library_name* **component metadata library. The name of the pack is** *pack_name***, and the name of the component is** *component_name***.**

Explanation: The modification level in the solution pack metadata library does not match the modification level in the component metadata library.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQM002E **The** *command_name* **line command is not valid: .**

Explanation: The specified line command is not valid.

System action: Processing continues.

User response: Specify a valid line command on the panel.

CCQO000W **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The PL/I XML parser issued the following exception warning code:** *code_number***.**

Explanation: While determining if the discover parameter metadata member is valid, the PL/I XML parser issued an exception warning code.

System action: Processing continues.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code.

CCQO001S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The PL/I XML parser issued the following exception error code:** *code_number***.**

Explanation: While determining if the Discover metadata member is valid, the PL/I XML parser issued an exception error code.

System action: Processing stops.

User response: See the *Enterprise PL/I for z/OS Programming Guide* for more information about the exception warning code. Contact IBM Software Support.

CCQO002S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The** *element_name* **element is unknown.**

Explanation: The specified element in the discover parameter metadata member is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO003S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. Content is not allowed for the** *element_name* **element, but content was found.**

Explanation: The specified element cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO004S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. Content is required for the** *element name* **element, but content was not found.**

Explanation: The specified element is missing required content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO005S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The content length for the** *element_name* **element cannot exceed** *maximum_number* **characters.**

Explanation: The specified element contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO006S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The** *element_name* **element cannot occur more than** *maximum_number* **times.**

Explanation: The specified element occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO007S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified element does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO008S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The** *attribute_name* **attribute in the** *element_name* **element cannot occur more than** *maximum_number* **times.**

Explanation: The specified attribute occurs too many times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO009S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The** *attribute_name* **attribute in the** *element_name* **element must occur at least** *minimum_number* **times.**

Explanation: The specified attribute does not occur enough times.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO010S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. Content is not allowed for the** *attribute_name* **attribute in the** *element_name* **element, but content was found.**

Explanation: The specified attribute cannot contain content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO011S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. Content is required for the** *attribute_name* **attribute in the** *element_name* **element, but content was not found.**

Explanation: The specified attribute requires content.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO012S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The content length for the** *attribute_name* **attribute in the** *element_name* **element in the cannot exceed** *maximum_number* **characters.**

Explanation: The specified attribute contains too many characters.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO013S **The XML structure of the** *member_name* **discover parameter metadata member is not valid. The** *attribute_name* **attribute in the** *element_name* **element is unknown.**

Explanation: The specified attribute is unknown.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO014S **The content of the *member_name* discover parameter metadata member is not valid because the value of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: A The specified value for an element in the discover parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO015S **The content of the *member_name* discover parameter metadata member is not valid because the value of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified value for an attribute in the discover parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO016S **The content of the *member_name* discover parameter metadata member is not valid because the data type of the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an element in the discover parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO017S **The content of the *member_name* product parameter metadata member is not valid because the data type of the *attribute_name* attribute in the *element_name* element is incorrect. The value is *value_name*.**

Explanation: The specified data type value for an attribute in the product parameter metadata member is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO050S **The *data_set_name* Discover REXX EXEC data set could not be initialized or was not found.**

Explanation: Tools Customizer could not find or could not initialize the specified Discover REXX EXEC data set.

System action: Processing stops.

User response: Ensure that the Discover REXX EXEC is specified correctly.

CCQO051W **The *data_sharing_group_ID* data sharing group ID cannot contain more than four characters.**

Explanation: The specified data sharing group ID contains too many characters.

System action: Processing continues.

User response: Ensure that the specified data sharing group ID does not exceed four characters.

CCQO052S **The *REXX_EXEC_name* Discover REXX EXEC was not found in the *data_set_name* Discover data set.**

Explanation: Tools Customizer could not find the Discover REXX EXEC in the specified data set.

System action: Processing stops.

User response: Ensure that the Discover data set was specified correctly.

CCQO053W **The *LPAR_name* LPAR name cannot contain more than eight characters.**

Explanation: The specified LPAR name contains too many characters.

System action: Processing continues.

User response: Ensure that the specified LPAR name does not exceed eight characters.

CCQO054W **The *subsystem_ID* DB2 SSID cannot contain more than four characters. The record was not processed.**

Explanation: The specified DB2 SSID contains too many characters.

System action: Processing continues.

User response: Ensure that the specified DB2 SSID does not exceed four characters.

CCQO055W **The *parameter_name* DB2 group attach name parameter is in the *record_name* Discover record, but a DB2 group attach name was not specified. The record was not processed.**

Explanation: The Discover record contains a data sharing group parameter, but a DB2 group attach name was not specified.

System action: Processing continues.

User response: Ensure that information is specified correctly on the Discover Customized Product Information panel.

CCQO056W The *parameter_name* DB2 parameter in the *record_name* Discover record did not have a DB2 group attach name or a DB2 SSID. The record was not processed.

Explanation: The Discover record did not have a DB2 group attach name or a DB2 subsystem ID in the DB2 parameter.

System action: Processing continues.

User response: Ensure that information is specified correctly on the Discover Customized Product Information panel.

CCQO057W The Discover EXEC could not find the *parameter_name* parameter in the metadata for the product to be customized. The record was not processed.

Explanation: The specified parameter could not be found in the metadata for the product to be customized.

System action: Processing continues.

User response: Ensure that information is specified correctly on the Discover Customized Product Information panel.

CCQO058W The *parameter_name* product parameter name in the *record_type* Discover record does not start with CCQ_LPR_, CCQ_DB2_, or CCQ_PRD_. The record was not processed.

Explanation: The parameter in the record does not start with CCQ_DB2_, CCQ_LPAR_, or CCQ_PRD_.

System action: Processing continues.

User response: Contact IBM Software Support.

CCQO059W The *parameter_name* product parameter cannot contain more than 72 characters. The record was not processed.

Explanation: The specified product parameter contains too many characters.

System action: Processing continues.

User response: Ensure that the specified product parameter does not exceed 72 characters.

CCQO060W The *record_name* Discover record from the REXX EXEC output must start with the following record type: *record_type*. The record was not processed.

Explanation: A Discover record from the REXX EXEC output must start with the specified DB2 record type.

System action: Processing continues.

User response: Contact IBM Software Support.

CCQO061I If you do not have a previously customized version of the product, do not run the Discover EXEC. Press END to go to the Customizer Workplace panel.

Explanation: This message is issued when you customize a product for a the first time. It prompts you to use the Discover EXEC to discover data from a previous customization of the specified product.

System action: Processing continues.

User response:

Tip: Using the Discover EXEC saves time and reduces errors that can error when parameters are specified manually. If you want to use the Discover EXEC, specify the required information on the Discover Customized Product Information panel. Otherwise, press End to continue without discovering data from a previous customization of the product.

CCQO062W The Discover EXEC could not find the following *parameter_name* parameter in the DB2 metadata. The record was not processed.

Explanation: The specified parameter is missing in the DB2 metadata.

System action: Processing continues.

User response: If this parameter is required, contact IBM Software Support.

CCQO064W The *Discover-record* Discover record did not have a parameter name. The record was not processed.

Explanation: A parameter name was missing in the Discover record.

System action: Processing continues.

User response: Contact IBM Software Support.

CCQO065W The value for the *parameter_name* parameter is ignored because it has more than *maximum_number* characters, which is the maximum length that is defined in the metadata. The value is *parameter_value*.

Explanation: The specified value exceeded the maximum allowed length, which was defined in the metadata. Tools Customizer truncated the extra characters.

System action: Processing continues.

User response: Contact IBM Software Support.

CCQO066W The *record_name* Discover record from the Discover REXX EXEC output does not have a parameter value. The record was not processed.

Explanation: The Discover record was missing a parameter value from the Discover EXEC output.

System action: Processing continues.

User response: Ensure that information was specified correctly on the Discover Customized Product Information panel.

CCQO067W The *parameter_name* parameter is defined in the metadata to support one value, but more than one value was found. The last value was used.

Explanation: The definition of the parameter in the metadata supports one value, but more than one value was specified. Only the last value was used.

System action: Processing continues.

User response: Ensure that information was specified correctly on the Discover Customized Product Information panel.

CCQO068W The value of the *parameter_name* parameter is ignored because the parameter is defined as internal=true. The value is *value_name*.

Explanation: The specified value of the parameter is ignored because it is defined as internal=true.

System action: Processing continues.

User response: Ensure that information was specified correctly on the Discover Customized Product Information panel.

CCQO069W The Discover EXEC did not find the *parameter_name* parameter in the LPAR metadata. The record was not processed.

Explanation: The specified parameter is missing from the LPAR metadata.

System action: Processing continues.

User response: Ensure that information was specified correctly on the Discover Customized Product Information panel.

CCQO070W The *record_type* Discover record contains an incorrect delimiter between the Environment section and the Data section. The record was not processed.

Explanation: Tools Customizer found an incorrect delimiter between the Environment section and the Data section.

System action: None.

User response: No action is required.

CCQO071W The *member_name* member could not be found in the *data_set_name* Discover data set.

Explanation: Tools Customizer could not find the specified Discover data set.

System action: None.

User response: No action is required.

CCQO072S The *member_name* discover metadata member was not found in the *data_set_name* metadata data set.

Explanation: Tools Customizer could not find the specified metadata member in the data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO073E The *member_name* discover metadata member is not valid because the default length for the *element_name* parameter element exceeds the length of the parameter. The default length is *default_length*, and the specified length is *specified_length*. The default length will be truncated accordingly.

Explanation: The default length for the specified parameter element is longer than the parameter.

System action: Processing continues.

User response: No action is required.

CCQO074S The content of the *member_name* discover metadata member is not valid. The value of the *attribute_name* attribute in the *element_name* element is not valid. The value of the attribute is *value_name*.

Explanation: The specified value is not valid.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO075W The *configuration_ID* configuration ID in the *record_name* Discover record is incorrect. The record was not processed.

Explanation: The specified configuration ID is not correct.

System action: Processing continues.

User response: No action is required.

CCQO076W **The *configuration_ID* configuration ID cannot contain more than *maximum_number* characters. The record was not processed.**

Explanation: The specified configuration ID contains too many characters.

System action: Processing continues.

User response: No action is required.

CCQO077S **The discover metadata member was not found in the *data_set_name* component data set that is part of the *data_set_name* pack.**

Explanation: The discover metadata member was not found in the specified component data set.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQO080I ***Product_name* does not support the Discover process.**

Explanation: The specified product does not support the Discover process.

System action: None.

User response: No action is required.

CCQP000E **The value of the *mode_name* DB2 mode is not valid for the *level_name* DB2 level.**

Explanation: The specified DB2 mode is not valid for the DB2 level.

System action: Processing stops.

User response: Specify a valid DB2 mode for the DB2 level.

CCQP001E **The value of the *mode_name* DB2 mode is missing.**

Explanation: The specified DB2 mode is not defined.

System action: Processing stops.

User response: Specify a value for the DB2 mode.

CCQP002E **The value of the *mode_name* DB2 level is missing.**

Explanation: The specified DB2 level is not defined.

System action: Processing stops.

User response: Specify a value for the DB2 level.

CCQP003E **The value of the *level_name* DB2 level is not valid.**

Explanation: The specified DB2 level does not have a valid name.

System action: Processing stops.

User response: Specify a valid value for the DB2 level.

CCQP004S **The *parameter_name* parameter does not exist in the CCQ$$DB2 DB2 parameter metadata member.**

Explanation: The CCQ$$DB2 DB2 parameter metadata member does not contain the specified parameter.

System action: Processing stops.

User response: Contact IBM Software Support.

CCQP005E **The value of the *subsystem_ID* DB2 SSID is missing.**

Explanation: The specified DB2 SSID is not defined.

System action: Processing stops.

User response: Specify a valid value for the DB2 SSID.

CCQP006E **The value of the *group_attach_name* DB2 group attach name is missing.**

Explanation: The specified DB2 group attach name is not defined.

System action: Processing stops.

User response: Specify a valid DB2 group attach name.

CCQQ000E **Specify a valid metadata library. Each qualifier of the library must start with an alphabetic character and must be 1-8 alphanumeric characters. The library name must be 1-44 characters.**

Explanation: The metadata library was not specified in the correct format. The high-level qualifier must contain alphanumeric characters, and the first character cannot be numeric. The name cannot contain wildcard characters, such as asterisks (*) and percent signs (%).

System action: Tools Customizer prompts for the correct library name.

User response: Specify a library name in the correct format.

CCQQ001E **The *data_set_name* data set name that was specified for the metadata library was not found.**

Explanation: The data set does not exist, or the data set name was written in the incorrect format. The

high-level qualifier must contain alphanumeric characters, and the first character cannot be numeric. The name cannot contain wildcard characters, such as asterisks (*) and percent signs (%).

System action: Tools Customizer prompts for the correct data set name.

User response: Specify a data set name in the correct format.

CCQQ002E **The data set name that was specified for the *library_name* metadata library cannot be opened.**

Explanation: Tools Customizer could not open the data set.

System action: Tools Customizer prompts for an available data set.

User response: Ensure that the specified data set is available for Tools Customizer to open it.

CCQQ003E **The *data_set_name* data set name that was specified for the metadata sample library is not valid. The data set must be in the following format: `HLQ.S`*xxx*`SAMP`.**

Explanation: The specified data set name was not specified in the correct format.

System action: None.

User response: Specify the data set name in the following format: `HLQ.S`*xxx*`SAMP`, where *xxx* is the three-character prefix for the product.

CCQQ004E **The *data_set_name* data set is being used by another user. Try again when the data set is not being used.**

Explanation: Another user is using the specified data set.

System action: None.

User response: Ensure that the specified data set is not being used.

CCQQ009E **The *data_set_name* data set name that was specified for the metadata library is not valid because the data set is empty.**

Explanation: The specified data set is empty.

System action: Tools Customizer prompts for an available data set.

User response: Ensure that the specified data set is available for Tools Customizer to open it.

CCQQ011E **The *library_name* metadata library for the component that is part of the *library_name* pack was not found in the catalog. The name of the pack is *pack_name*, and the name of the component is *component_name*.**

Explanation: The specified metadata library is not in the catalog.

System action: None.

User response: Specify another metadata library.

CCQQ012E **The *library_name* metadata library for the component that is part of the *library_name* pack cannot be opened.**

Explanation: The specified metadata library cannot be opened.

System action: None.

User response: Ensure that the name of the library is specified correctly.

CCQS000I **Tools Customizer is being invoked for the first time or the previous ISPF session ended before Tools Customizer was exited. In both cases, the fields on this panel are populated with default values. Review these default values or specify new values to be used to customize products or packs.**

Explanation: When you customize a stand-alone product or a solution pack for the first time, or when an ISPF session unexpectedly ends before the ISPF profile is saved, you must specify or review your Tools Customizer user settings.

System action: Processing stops.

User response: Review and accept the default settings, or specify new settings.

CCQS001E **The following command is not valid:** *command_name.*

Explanation: The specified command is not a valid command on the panel.

System action: Processing stops.

User response: Specify a valid command.

CCQS002W **The *data_set_name* Discover data set could not be found.**

Explanation: Tools Customizer could not find the specified data set.

System action: Processing continues.

User response: Ensure that the data set name is specified correctly.

CCQS003W The *data_set_name* Discover data set was not found so it was created.

Explanation: Tools Customizer could not find the specified data set.

System action: Processing continues.

User response: Ensure that the data set name is specified correctly.

CCQS004I The settings were saved.

Explanation: The settings that you changed were saved.

System action: Processing continues.

User response: No action is required.

CCQS006W The length of a qualifier for the *data_set_name* customization library data set exceeds 26 characters.

Explanation: The qualifier for the customization library data set is too long. The qualifier cannot exceed 26 characters.

System action: Processing continues.

User response: Specify a qualifier that is 26 characters or less.

CCQS007E The discover data set *data_set_name* could not be opened with the *option-type* option.

Explanation: The specified option could not open the Discover data set.

System action: None.

User response: Specify a data set to which you have WRITE access.

CCQS008E An error occurred while the *data_set_name* Discover data set was being created.

Explanation: While the specified data set was being created, an error occurred.

System action: Processing continues.

User response: Ensure that you have WRITE authority access to this data set.

CCQS010E The customization library qualifier is not valid.

Explanation: The customization library qualifier that was specified is not valid.

System action: None.

User response: Specify a valid qualifier for the customization library.

CCQS011E The group attach option is not valid.

Explanation: The group attach option that was specified is not valid.

System action: None.

User response: Specify a valid option for the group attach option.

CCQS012E The Tools Customizer metadata library is not valid.

Explanation: The metadata library that was specified is not a valid data set.

System action: None.

User response: Specify a valid data set for the metadata library.

CCQS013E The Discover data set is not valid.

Explanation: The Discover data set that was specified is not a valid data set.

System action: None.

User response: Specify a valid Discover data set.

CCQS014E The data store data set is not valid.

Explanation: The data set that was specified is not a valid data set.

System action: None.

User response: Specify a valid data store data set.

CCQS015E Tools Customizer is already running.

Explanation: A session of Tools Customizer is already running in your environment. Only one Tools Customizer session is allowed.

System action: None.

User response: The trace data set is being used. Free the trace data set, and start Tools Customizer again.

CCQS018E Information on the first line of the job card exceeds 57 characters.

Explanation: The first line of the job card can contain only 57 characters. This character limit includes a continuation character.

System action: Tools Customizer clears the first line of the job card.

User response: Specify information that does not exceed 57 characters on the first line of the job card.

CCQS019E The required trace data set, *data_set_name*, is currently not accessible.

Explanation: The trace data set must be accessible.

System action: Processing stops.

User response: Ensure that the trace data set is accessible.

CCQS020E An error occurred while the customization library data set was being created. ALTER authority on the high-level qualifier for the customization library data set is required.

Explanation: To create the customization library data set, ALTER authority on the specified high-level qualifier must be granted.

System action: None.

User response: Ensure that ALTER authority for the specified customization library data set is granted.

CCQS021E The value *value_name* in the field that contains the cursor position is not valid.

Explanation: The specified value is not valid.

System action: None.

User response: Specify a valid value.

CCQS022E An error occurred while the customization library data set was being opened. UPDATE authority on the high-level qualifier for the customization library data set is required.

Explanation: To open the customization library data set, UPDATE authority on the specified high-level qualifier must be granted.

System action: None.

User response: Ensure that UPDATE authority for the specified customization library data set is granted.

CCQS023E An error occurred while the customization library data set was being opened. UPDATE authority on the high-level qualifier for the customization library data set is required.

Explanation: To open the customization library data set, UPDATE authority on the specified high-level qualifier must be granted.

System action: None.

User response: Ensure that UPDATE authority for the specified customization library data set is granted, or specify a different high-level qualifier for the customization library data set on the Tools Customizer Settings panel.

CCQS024E An error occurred while the customization library data set was being created. ALTER authority on the high-level qualifier for the customization library data set is required.

Explanation: To create the customization library data set, ALTER authority on the specified high-level qualifier must be granted.

System action: None.

User response: Ensure that ALTER authority for the specified customization library data set is granted, or specify a different high-level qualifier for the customization library data set on the Tools Customizer Settings panel.

CCQS030E The following command is not a valid CREATE statement: *command_statement*.

Explanation: The specified CREATE command statement is invalid because it contains blanks or alphabetic characters.

System action: Processing stops.

User response: Specify a valid CREATE command statement. The correct syntax is CREATE *nn*, where *nn* is 1 - 99.

CCQS031E The following command is not a valid CREATE statement: *command_statement*. The number that can be specified with the CREATE command is 1 - 99.

Explanation: The specified CREATE command statement is invalid because it contains either 0 or a number greater than 99.

System action: Processing stops.

User response: Specify a valid CREATE command

statement. The correct syntax is CREATE *nn*, where *nn* is 1 - 99.

CCQT000I **The product configuration ID** *copied_configuration_ID* **was successfully copied from** *configuration_ID*.

Explanation: The specified configuration ID was copied.

System action: None.

User response: No action is required.

CCQT001E **The** *command_name* **line command was specified more than once, which is not allowed.**

Explanation: The specified line command cannot be specified more than one time.

System action: Processing stops.

User response: Specify the line command only once.

CCQT002E **The** *configuration_ID* **configuration ID already exists. Specify a different configuration ID.**

Explanation: The specified configuration ID exists.

System action: Processing stops.

User response: Ensure that the specified configuration ID is unique.

CCQT003I **The product configuration ID** *configuration_ID* **was created.**

Explanation: The specified configuration ID was created.

System action: None.

User response: No action is required.

CCQT004I **The product configuration ID** *configuration_ID* **was removed.**

Explanation: The specified configuration ID was removed.

System action: None.

User response: No action is required.

CCQT005E **The product configuration ID** *configuration_ID* **is not valid. The product configuration ID cannot contain a colon (:).**

Explanation: The specified configuration ID contains a colon (:), but a colon is not valid.

System action: Processing stops.

User response: Specify a configuration ID that does not contain a colon.

CCQT006E **The** *configuration_ID* **configuration ID exists. Specify a different configuration ID.**

Explanation: The specified configuration ID exists.

System action: Processing stops.

User response: Specify another configuration ID.

CCQT007E **The** *configuration_ID* **configuration ID exists but was removed from the list of configurations. To use this configuration ID, you must restore it.**

Explanation: The specified configuration ID exists but was removed from the list of available configuration.

System action: Processing stops.

User response: Specify another configuration ID. To restore the specified configuration ID, issue the CREATE command, and specify the same configuration ID again.

CCQT008E **The** *configuration_ID* **configuration ID exceeds** *maximum_number* **characters.**

Explanation: The specified configuration ID contains too many characters.

System action: Processing stops.

User response: Specify another configuration ID that does not exceed the maximum number of characters that was set by DB2 UET.

CCQT010I **Create request for** *configuration_ID* **configuration was cancelled by user.**

Explanation: The request to create the specified configuration was canceled.

System action: Processing stops.

User response: No action is required.

CCQT011I **The** *configuration_ID* **configuration was not copied.**

Explanation: The specified configuration was not copied.

System action: Processing stops.

User response: No action is required.

CCQT012I **The *configuration_ID* configuration was not removed.**

Explanation: The specified configuration was not removed.

System action: Processing stops.

User response: No action is required.

CCQT013I **None of the configurations were copied or removed. All of the previously selected configurations are deselected.**

Explanation: The selected configurations were not copied or removed, and they are deselected.

System action: Processing stops.

User response: No action is required.

CCQT014E **Specify Y or N and press Enter to continue, or press End to cancel.**

Explanation: A function requires input.

System action: Processing stops.

User response: To continue, specify Y or N and press Enter. Otherwise, press End to cancel.

CCQT015E **The *command_name* command is not allowed during the process of "Select" configuration line command.**

Explanation: The specified command is not allowed while the line command for selecting configurations is processing.

System action: Processing stops.

User response: Remove the specified line command.

CCQT016I **The *configuration_ID* configuration was not created**

Explanation: The specified configuration was not created.

System action: Processing stops.

User response: No action is required.

CCQT017I **The *configuration_ID* configuration was not copied.**

Explanation: The specified configuration was not copied.

System action: Processing stops.

User response: No action is required.

CCQT018E **Specify Y or N, and press Enter.**

Explanation: A function requires input.

System action: Processing stops.

User response: To continue, specify Y or N, and press Enter.

CCQT019I **The select *configuration_ID* configuration process ended.**

Explanation: The select process for the specified configuration is finished.

System action: Processing stops.

User response: No action is required.

CCQT020E **The *configuration_ID* configuration was not created because the data store was not accessible.**

Explanation: The specified configuration was not created because the data store could not be accessed.

System action: Processing stops.

User response: Ensure that the data store is accessible and create the configuration again.

CCQT021E **The *configuration_ID* configuration was not copied because the data store was not accessible.**

Explanation: The specified configuration was not copied because the data store could not be accessed.

System action: Processing stops.

User response: Ensure that the data store is accessible and copy the configuration again.

CCQT025I **The *configuration_ID* configuration was not updated.**

Explanation: The specified configuration was not updated because the edit process was canceled.

System action: Processing stops.

User response: No action is required.

CCQT027I **The product configuration was successfully updated.**

Explanation: The configuration was updated.

System action: Processing continue.

User response: No action is required.

CCQX001S *Product_name* **has already been customized by using values from** *data_set_name* **data store data set. Switch to the specified data store data set to continue customizing this product.**

Explanation: The specified product was customized by using values from the specified data store data set.

System action: Processing stops.

User response: Use the specified data store data set to continue customizing the product.

CCQX002S *component_name* **has already been customized by using values from** *data_set_name* **data store data set. Switch to the specified data store data set to continue customizing this component.**

Explanation: The specified component was customized by using values from the specified data store data set.

System action: Processing stops.

User response: Use the specified data store data set to continue customizing the component.

CCQX011I *Product_name* **was not found.**

Explanation: The specified product was not found.

System action: Processing stops.

User response: Specify another product.

DB2 Utilities Enhancement Tool messages

Look up DB2 UET messages to obtain information about them, including message explanations and suggested responses.

Each message has a unique message ID. The first three to four letters of an ID indicate the DB2 UET component for which the message was issued.

- *ABP* indicates the ISPF interface.
- *ABPB* indicates the batch interface.
- *ABPG* is for global messages that pertain to multiple components.
- *ABPM* indicates the DB2 UET maintenance utility (ABPMAINT).
- *ABPP* indicates a parser component. (These messages are primarily for use by Software Support.)
- *ABPS* indicates the started task.
- *ABPU* indicates the DSNUTILB intercept.

Also, all message IDs have a severity code as the last character, as follows:

- I: Information only. No user action is required.
- W: Warning message. Results might not be as expected.
- E: Error message. Some errors might be user-correctable. Read the User Response to determine the appropriate course of action.
- S: Severe error message. A severe internal or environmental error occurred. Usually, users need to contact Software Support for assistance in resolving these errors.

In the messages output, a timestamp is often displayed after the message identifier and before the message text to indicate when the message was issued. The timestamp is composed of a Julian date followed by a time in the format HH:MM:SS:tt (where *HH* is hours, *MM* is minutes, *SS* is seconds, and *tt* is hundredths of a second). This timestamp does not occur in messages that are issued from the ISPF interface or batch interface (*ABP* or *ABPB* messages) or in any messages that are issued as WTO messages. (The WTO messages include a system timestamp instead.)

Related concepts:

"Determining whether thread blocking and canceling occurred" on page 234
DB2 UET generates several types of output for a thread-cancel or thread-blocker

job step. Check this output to determine if your job step completed successfully.

"Determining whether the REORG TABLESPACE utility enhancements were implemented" on page 293
To determine whether DB2 UET was able to generate the mapping table and mapping table index for the REORG TABLESPACE utility, check the DB2 messages in the SYSPRINT data set for the utility. The DB2 UET SYSPRINT displays message ABPU5407I to indicate that the mapping table was created successfully.

"Determining whether thread blocking and canceling occurred" on page 184
To determine whether thread blocking and cancelation processing occurred, you can check the DB2 UET messages in the SYSPRINT data set for the DB2 utility. Also, you can review the batch reports on threads canceled. Use SDSF or an equivalent tool to view this information.

ABP011E *ispf_user* **is not authorized to view the selected panel**

Explanation: The specified user cannot view a DB2 Utilities Enhancement Tool panel because the user does not have the proper authority to do so under the security requirements for the product.

User response: If the user requires access to the panel, ask the product administrator or security administrator to provide the user with the required level of authority.

ABP012E *ispf_user* **is not authorized to save system information**

Explanation: The specified user cannot save changes to options on the Control System panel because the user does not have the authority to make these changes.

User response: If the user requires these changes, contact the product administrator for assistance.

ABP013E **ABP API Error:** *API* **- Return Code:** *number* **- Reason Code:** *number*

Explanation: The DB2 Utilities Enhancement Tool ISPF interface encountered an error when communicating with an internal application programming interface (API). This message contains the text of the error message.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABP014E **Status information cannot be displayed before thread is canceled.**

Explanation: You cannot display the Cancel Thread Information panel for a thread because the thread has not been canceled.

User response: On the Thread Summary Report panel, specify the I(Info) line command next to a thread for which CAN or ERR is displayed in the **Msg** column. The Cancel Thread Information panel will be displayed and present messages from cancelation processing.

ABP015I **Cancel command was issued successfully**

Explanation: A command for canceling a DB2 thread was successfully issued from the DB2 Utilities Enhancement Tool ISPF interface. This command can be for a standard DB2 cancelation (the DB2 -CANCEL THREAD command) or for an escalated cancelation.

User response: No action is required.

ABP016E **Cancel command returned error**

Explanation: During the processing of a cancel command, an error occurred. The command can be for a standard DB2 cancelation (the DB2 -CANCEL THREAD command) or an escalated cancelation.

User response: Review the related error messages to determine the cause of the error. Then correct the error and attempt the cancelation again.

ABP017E **Access not allowed if no DB2 object is specified**

Explanation: When creating a thread filter on the Specify Thread Filter Criteria panel, you specified a value in the **Access** field without also specifying the DB2 objects to which the access level applies, as required.

User response: For the **Access** value to be used for thread filtering, you must also specify a value in at least one of the following fields: **Database**, **Table space**, **Partition**, **Table**, **Synonym**, **Alias**, **View**, or **Index**, or a **Creator** field.

ABP018E **Partition not allowed if Database and Table space not specified**

Explanation: When creating a thread filter on the Specify Thread Filter Criteria panel, you specified a value in the **Partition** field without also specifying the database or table space in which the partition occurs.

User response: For the **Partition** value to be used for thread filtering, you must also specify a value in one or both of the following fields: **Database** and **Table space**.

ABP019E Command not allowed when Thread Token is zero

Explanation: On the Thread Summary Report panel, you attempted to specify the `C` (Cancel) or `O` (Objects) line command for a thread that has a thread token value of zero. You cannot specify these commands for threads with a token value of zero.

User response: No action is required.

ABP020E Cancel Thread must be set to Yes if Escalated Cancel is set to Yes

Explanation: The value **Yes** was specified in the **Escalated cancel** field but not in the **Cancel thread** field. You must specify **Yes** in both of these fields to perform an escalated cancelation of the thread.

User response: Type **Yes** in the **Cancel thread** field as well as in the **Escalated cancel** field, and then press Enter to perform an escalated cancelation of the thread.

ABP030E Sign on to started task ABPID denied by security exit for user *USERID*.

Explanation: The DB2 Utilities Enhancement Tool security exit has denied the user sign-on access to the DB2 Utilities Enhancement Tool started task.

User response: Contact the DB2 Utilities Enhancement Tool administrator.

ABP031E User *USERID* is not currently signed on.

Explanation: Navigation through the ISPF interface panels is severely limited when running the interface with no sign-on. This situation occurs when sign-on is denied by security exit or the ABPID saved in an ISPF variable is not available at sign-on.

User response: Contact the DB2 Utilities Enhancement Tool administrator.

ABP032E Thread is no longer active.

Explanation: The selected thread is no longer active.

User response: No action is required.

ABP033E DB2 subsystem *SSID* unavailable. Authorization error.

Explanation: DB2 subsystem is unavailable due to an authorization error.

User response: Select a different DB2 SSID or contact your DB2 administrator.

ABP034E DB2 subsystem *SSID* unavailable. Plan not bound.

Explanation: DB2 subsystem is unavailable because the DB2 Utilities Enhancement Tool plan is not bound.

User response: Select a different DB2 SSID or bind the DB2 Utilities Enhancement Tool plan.

ABP035E A DB2 connection is required for this option.

Explanation: No DB2 connection exists.

User response: On panel ABPLPRF1 (Set DB2 System), select an appropriate DB2 SSID from the list.

ABP036E SQL error code *SQLCODE* encountered in DB2 Utilities Enhancement Tool started task ABPID.

Explanation: An SQL error has occurred.

User response: Contact the DB2 Utilities Enhancement Tool administrator. Messages in the DB2 Utilities Enhancement Tool server SYSPRINT might be useful in determining the error.

ABP037E ABP started task ABPID is not currently accepting requests.

Explanation: The connection to the DB2 Utilities Enhancement Tool server has been lost.

User response: Contact the DB2 Utilities Enhancement Tool administrator.

ABP038E The connection to *SSID* has been lost.

Explanation: The connection to the DB2 subsystem has been lost.

User response: Contact the DB2 Utilities Enhancement Tool administrator.

ABP039E DB2 subsystem *SSID* unavailable. Subsystem is not running.

Explanation: The DB2 subsystem *SSID* saved in the ISPF profile used to connect the ISPF session is not running.

User response: Select a different SSID from panel ABPLPRF1 (Set DB2 System) or contact the DB2 Utilities Enhancement Tool administrator.

ABP040E DB2 subsystem *SSID* unavailable. Access is restricted.

Explanation: Access to selected DB2 subsystem is restricted.

User response: Contact the DB2 Utilities Enhancement Tool administrator.

ABP041E Session has been terminated by the server.

Explanation: The server ended the session.

User response: Contact the DB2 Utilities Enhancement Tool administrator.

ABP042E Partition number out of range.

Explanation: Partition number out of range on panel ABPLTHD1

User response: Enter a valid partition value in the range of 1 through 4096.

ABP043E DB2 Utilities Enhancement Tool interface version does not match selected server version.

Explanation: The instance version that you selected is different than the ISPF interface version that is currently running.

User response: Select an instance version that matches the ISPF interface version or run an ISPF interface that matches the required server instance.

ABP044E Port number out of range.

Explanation: The DDF IP port number value is out of range.

User response: Enter a valid port number in the range of 1 through 32767.

ABP045E Invalid connection type.

Explanation: The ISPF connection type number is out of range.

User response: Enter a valid connection type number in the range of 1 through 32767.

ABP101E ISPF error: *error_message*

Explanation: An ISPF error occurred while the DB2 Utilities Enhancement Tool ISPF user interface was running.

User response: Restart the DB2 Utilities Enhancement Tool ISPF interface. If you continue to receive this error, contact IBM Software Support.

ABP102E Invalid Command. Enter a valid command

Explanation: An invalid primary command was specified at the command line on a panel, or an invalid value was specified for a menu option.

User response: Specify a valid primary command or menu option. See the panel-level Help for descriptions of the valid primary commands or menu options. To display Help information, press F1.

ABP103E Invalid Line Command. Enter a valid line command

Explanation: An invalid line command was specified in the **C** column on a DB2 Utilities Enhancement Tool ISPF panel.

User response: Specify a valid line command. See the panel-level Help for descriptions of the valid line commands for the panel. To display Help information, press F1.

ABP105E Invalid value

Explanation: An invalid value was specified in a field on a DB2 Utilities Enhancement Tool ISPF panel.

User response: Specify a valid value in the field. See the field-level or panel-level Help for descriptions of the valid values. To display Help information, press F1.

ABP107E Result not found

Explanation: You specified the FIND primary command to locate an item on a panel that contains a selection list. However, no item matched the FIND criteria.

User response: Check the FIND command to ensure that it was correctly specified. If necessary, correct the command and reissue it. If the command is correct, but the item was not found, it does not exist on the panel.

ABP900I The Discover EXEC was successful

Explanation: The DISCOVER process ended successfully.

User response: No action is required.

ABP901E Options module *library_name(module_name)* **was not found**

Explanation: The DISCOVER process did not find the initialization options member in the previous installation location.

User response: Specify the correct options module name in the SAMP library of the previous installation.

ABP902E An options template name was specified instead of an actual options module name

Explanation: An options module template name was specified instead of an initialization options module.

User response: Specify the correct options module name in the SAMP library of the previous installation.

ABP903E DB2 UET version 2.2 installation SAMPLIB *library_name* not found

Explanation: The DISCOVER process did not find the SAMP library of the installation that you are customizing.

User response: Specify the correct SAMP library name for the installation.

ABP904E POLICY member not found

Explanation: Copying the POLICY member from the previous installation was specified for use in the current installation. However, the DISCOVER process did not find the specified POLICY member *library_name*(*member_name*) in the SAMP library of the previous installation.

User response: Specify the correct name for the POLICY member from the previous installation.

ABP905E DB2 UET version 2.1 installation SAMPLIB *library_name* not found

Explanation: The DISCOVER process did not find the SAMP library of the previous installation.

User response: Specify the correct SAMP library name for the previous installation.

ABPB6020I ABPPARMS input parameters are as follows:

Explanation: This message introduces the list of ABPPARMS input parameters that are specified for the job. These parameters are included in the message that immediately follows this one.

User response: No action is required.

ABPB6021E The PARM value specified on the EXEC statement is invalid; delimiter not found

Explanation: The DB2 Utilities Enhancement Tool batch program failed to initialize because the PARM in the EXEC statement does not include the required delimiter between the *blocker_id* value and the *action* value.

User response: Add a comma as the delimiter between the *blocker_id* value and the *action* value in the PARM, without any blank spaces between these values. For more information about the correct syntax for thread blocking job steps, see the batch interface section in the *DB2 Utilities Enhancement Tool for z/OS User's Guide*.

ABPB6022E The PARM value specified on the EXEC statement is invalid: BLOCKER_ID value not found

Explanation: The DB2 Utilities Enhancement Tool batch program failed to initialize because the PARM in the EXEC statement does not include a *blocker_id* value. A *blocker_id* is a user-defined identifier that must be specified for a thread-blocker function.

User response: Add a unique *blocker_id* value to the PARM. This value can be up to 32 characters long. For more information about the correct syntax, see the batch interface section in the *DB2 Utilities Enhancement Tool for z/OS User's Guide*.

ABPB6023E The PARM value specified on the EXEC statement is invalid: BLOCKER_ID value is too long

Explanation: The DB2 Utilities Enhancement Tool batch program failed to initialize because the PARM in the EXEC statement contains a *blocker_id* that is longer than the maximum number of characters that is allowed for this value.

User response: In the PARM, specify a *blocker_id* value that is no longer than 32 characters in length. For more information about the correct syntax, see the batch interface section in the product documentation.

ABPB6024E The PARM value specified on the EXEC statement is invalid: BLOCKER_ID value contains the invalid character, '*single_character*'

Explanation: The DB2 Utilities Enhancement Tool batch program failed to initialize because the PARM in the EXEC statement contains a *blocker_id* value wit h an invalid character. The following characters are invalid: a blank space, a single quotation mark, and a comma (other than the comma that is used as the delimiter).

User response: Correct the *blocker_id* value in the PARM. Ensure that this value does not contain a blank space, single quotation mark, or comma. For more information about the correct syntax for thread blocking, see the batch interface section in the product documentation.

ABPB6025E The PARM value specified on the EXEC statement is invalid: ACTION value not found

Explanation: The DB2 Utilities Enhancement Tool batch program failed to initialize because the PARM in the EXEC statement does not include an *action* value. An *action* value is required to identify the type of thread-blocker action.

User response: Add one of the following valid action values to the PARM: BLOCK_THREADS to prevent new threads from forming; ALLOW_THREADS to end

thread blocking and allow new threads to form again; or DELETE_BLOCKER_ID to remove all data associated with the blocker ID from the table for thread-blocker information.

ABPB6026E The PARM value specified on the EXEC statement is invalid: ACTION value not recognized

Explanation: The DB2 Utilities Enhancement Tool batch program failed to initialize because the PARM in the EXEC statement contains an invalid *action* value. The *action* value identifies the type of thread-blocker function.

User response: Specify a valid *action* value in the PARM. Valid values are: BLOCK_THREADS, ALLOW_THREADS, and DELETE_BLOCKER_ID. For more information about the correct syntax for thread blocking, see the batch interface section in the product documentation.

ABPB6030I EXEC PARM value is: *blocker_id,action*

Explanation: DB2 Utilities Enhancement Tool has begun processing the PARM in the EXEC statement of the batch job step. This message indicates the *blocker_id* and *action* values that are specified for the PARM. The PARM is required to implement thread blocking.

User response: No action is required.

ABPB6031I Thread blocker BLOCKER_ID is: *blocker_id*

Explanation: This message provides the *blocker_id* value that was specified for the PARM in the EXEC statement of the DB2 Utilities Enhancement Tool batch job step. The *blocker_id* is a user-defined identifier that must be specified for a thread-blocker function.

User response: No action is required.

ABPB6032I Thread blocker ACTION is: *action*

Explanation: This message identifies the thread-blocker action that was specified for the PARM in the EXEC statement of the DB2 Utilities Enhancement Tool batch job step. The *action* value can be one of the following:

- BLOCK_THREADS prevents new threads from forming
- ALLOW_THREADS ends thread blocking and allows new threads to form again
- DELETE_BLOCKER_ID removes all data for a blocker ID from the DB2 table that stores thread-blocker information

User response: No action is required.

ABPB6040I Processing of input parameters has started

Explanation: The processing of the ABPPARMS input parameters has started. The syntax and validity of each parameter will be evaluated.

User response: No action is required.

ABPB6041I Processing of input parameters ended with an error. The job terminated.

Explanation: While processing the ABPPARMS input parameters, DB2 Utilities Enhancement Tool encountered an error within the input control cards. As a result, the thread-cancelation job ended.

User response: See the additional messages that were issued for this error to obtain information that can help you locate and diagnose the problem with the input control cards. Then correct the problem and resubmit the job.

ABPB6042I Processing of input parameters ended successfully

Explanation: All of the ABPPARMS input parameters were processed successfully. The parameters passed syntax and validity checking. As a result, the thread-cancelation job continues.

User response: No action is required.

ABPB6050I Initialization is complete.
DB2SSID=*db2_ssid* **DB2REL=***db2_version*
NETID=*network_name*
LUNAME=*logical_unit_name*
LOCNAME=*location_name*

Explanation: Initialization processing completed. This message identifies the DB2 subsystem identifier, the DB2 version and release, the network identifier, the LU name, and the local location name.

User response: No action is required.

ABPB6060I Thread blocker processing has started: *thread_blocker_action*

Explanation: DB2 Utilities Enhancement Tool has started thread-blocker processing. This message indicates the type of thread-blocker action this is being performed.

User response: No action is required.

ABPB6061I Thread blocker processing ended.
Highest RC=*return_code*

Explanation: The thread blocker action that you specified in the PARM of the EXEC statement completed. This message provides the highest return code from thread blocker processing.

User response: If the return code is 0, the thread blocker processing completed successfully. If the return code is greater than 4, an error might have occurred. For more information, look up the return code in the DB2 Utilities Enhancement Tool documentation. You can also view the messages in the job output to determine why the return code was issued.

ABPB6062W The cancel request will not block threads because it does not specify a parameter for a DB2 object: Request=*request_number*

Explanation: The specified cancel request does not include a parameter for a DB2 object. Consequently, DB2 Utilities Enhancement Tool cannot determine the set of objects for which to block new threads. Therefore, no thread blocking will occur.

User response: To block threads, you must specify at least one parameter for a DB2 object in the cancel request. If you do not want to block threads for this cancel request, no action is necessary.

ABPB6064E Error occurred while processing the thread-blocker action *action_name*

Explanation: An error occurred while DB2 Utilities Enhancement Tool was attempting to perform the specified thread-blocker action.

User response: To diagnose the error, check the reports that were generated for the batch job step and check the DB2 Utilities Enhancement Tool started task logs. If you need assistance, contact IBM Software Support.

ABPB6100I DB2 Utilities Enhancement Tool *product_release* **execution has started**

Explanation: DB2 Utilities Enhancement Tool has started. This message indicates the version and release of the product configuration that is running.

User response: No action is required.

ABPB6101I Execution type is: *batch_exec_type*

Explanation: The DB2 Utilities Enhancement Tool batch job is running in the execution mode that is specified. The execution mode can be: EXECUTE (an actual run), SIMULATE (a trial run), or CHECKPARMS (a check of the batch syntax only). The execution mode is set by the EXEC_TYPE parameter in the batch job.

User response: No action is required.

ABPB6120I Processing CANCEL_THREADS request=*request_number***, Started at** *time_stamp***; Parameters are as follows:**

Explanation: The processing of the specified CANCEL_THREADS request has started. This message displays the 4-digit request number that DB2 Utilities Enhancement Tool assigned to the request and the time when processing started. The time is in the format HH:MM:SS.s, where *HH* is hours, *MM* is minutes, *SS* is seconds , and *s* is tenths of a second. This message also introduces the list of parameters for the CANCEL_THREADS request, which are included in a separate message.

User response: No action is required.

ABPB6121I Processing of CANCEL_THREADS request=*request_number* **has ended at** *time_stamp_HH:MM:SS.s***: Threads canceled=***cancel_count***, RC=***return_code*

Explanation: The specified CANCEL_THREADS request ended. This message presents the request number, the time when the request ended, the number of threads that were canceled, and the return code. The time is in the format HH:MM:SS.s, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *s* is tenths of a second. The request number is the 4-digit sequence number that DB2 Utilities Enhancement Tool assigned to the request when the request started.

User response: No action is required.

ABPB6122I Processing of CANCEL_THREADS request=*request_number* **failed at** *time_stamp_HH:MM:SS.s***: RC=***return_code*

Explanation: The specified CANCEL_THREADS request ended in error. This message presents the request number, the time when the request processing failed, and the return code. The time is in the format HH:MM:SS.s, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *s* is tenths of a second. The request number is a 4-digit number that DB2 Utilities Enhancement Tool assigns to a CANCEL_THREADS request when it starts.

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Then correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6200E API Initialization failed

Explanation: The batch interface program failed to complete initialization. The failure occurred during initialization of the internal API.

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6201E API termination completed with errors

Explanation: At least one error occurred while terminating the batch interface program. The program could not stop the internal application programming interface (API).

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6202E Unable to locate a product configuration with ABPID=*identifier*

Explanation: The batch interface program failed to complete initialization because it could not locate the DB2 Utilities Enhancement Tool configuration that has the specified ABPID value.

User response: Check the value that is specified for the ABPID global parameter in the ABPPARMS DD statement. Ensure that the value correctly identifies an active configuration of the DB2 Utilities Enhancement Tool.

ABPB6203E Session creation failed RC=*return_code***, RSN=***reason_code***, Reason=***description*

Explanation: The batch interface program failed to complete initialization. The failure occurred during the creation of the batch interface session.

User response: To determine the cause of the failure, review the reason description in this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6204E Session termination completed with errors

Explanation: At least one error occurred during the termination of the batch interface program. The batch program could not end its session.

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6205E Error connecting to DB2 with DB2 SSID=*db2_ssid*

Explanation: The batch interface program failed to complete initialization. An error occurred while DB2 Utilities Enhancement Tool was attempting to connect to the specified DB2 subsystem.

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6206E Error disconnecting from DB2 with DB2 SSID=*db2_ssid*

Explanation: One or more errors occurred during the termination of the batch interface program. The program could not disconnect from the specified DB2 subsystem.

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPB6207E An ABPID is required in the ABPPARMS DD because a thread blocker action is specified

Explanation: You must specify a value for the ABPID global parameter in the ABPPARMS DD statement if you specify a thread-blocker action on the EXEC PARM.

User response: Ensure that a valid ABPID value is specified in the ABPPARMS DD statement if you specify a thread-blocker action on the EXEC PARM.

ABPB6208E A DB2 SSID is required in the ABPPARMS DD

Explanation: You must specify a value for the DB2SSID global parameter in the ABPPARMS DD statement.

User response: Specify a valid SSID for the DB2SSID parameter in the ABPPARMS DD statement.

ABPB6209E The *parameter_name* value specified in the ABPPARMS DD is not supported. Value=*parameter_value*

Explanation: The value specified for the named parameter in the ABPPARMS DD statement is not supported.

User response: Specify a valid value for the parameter in the ABPPARMS DD statement. See the DB2 Utilities Enhancement Tool documentation for the valid values.

ABPB6210E At least one CANCEL_THREADS command is required in the ABPPARMS DD.

Explanation: No CANCEL_THREADS command is specified in the ABPPARMS DD statement. At least one CANCEL_THREADS command is required.

User response: Specify one or more CANCEL_THREADS commands in the ABPPARMS DD statement.

ABPB6211I Setting retry count to zero since the retry interval is set to zero in the ABPPARMS DD for CANCEL_THREADS request=*request_number*

Explanation: A value of zero was specified for the CHECK_THDTERM_RETRY_INTERVAL parameter in the specified CANCEL_THREADS request. Consequently, the CHECK_THDTERM_RETRY_COUNT parameter was programmatically set to zero. These parameters control the frequency and number of times that DB2 Utilities Enhancement Tool checks whether a canceled thread has actually terminated in DB2.

User response: If you want the retry count to be a positive integer, you must specify a non-zero value for the retry interval parameter in the specified CANCEL_THREADS request.

ABPB6212E No DB2 object is specified to correspond with the ACCESS parameter for CANCEL_THREADS request=*request_number*

Explanation: To use the ACCESS parameter to filter threads, you must also specify parameters that identify the DB2 objects that the threads are referencing.

User response: Specify at least one parameter that identifies the DB2 object or objects that the threads are referencing, for example, TABLE or TABLESPACE. You can then filter threads based on their level of access (Read, Write, or All) to those objects.

ABPB6213E No DB2 Utilities Enhancement Tool configuration available for DB2 subsystem, DB2SSID=*db2_ssid*

Explanation: The batch interface program did not complete initialization because it could not locate a DB2 Utilities Enhancement Tool configuration that can connect to the specified DB2 subsystem.

User response: Make sure that the DB2 plan that DB2 Utilities Enhancement Tool uses is bound on the DB2 subsystem that is identified by the DB2SSID parameter in the ABPPARMS DD statement for at least one active DB2 Utilities Enhancement Tool configuration.

ABPB6214E No DB2 Utilities Enhancement Tool configuration is available, RC=*return_code***, RSN=***reason_code*

Explanation: The batch interface program did not complete initialization because it could not locate a DB2 Utilities Enhancement Tool configuration.

User response: Make sure that at least one DB2 Utilities Enhancement Tool configuration is active. If you continue to receive this error, contact IBM Software Support.

ABPB6215E Session has been terminated by the server.

Explanation: The batch interface program did not complete because the server stopped the session.

User response: Check with the system administrator to determine the reason for the termination of the batch interface program.

ABPB6250E Failed to locate active threads on the DB2 subsystem, DB2SSID=*db2_subsystem_id*

Explanation: DB2 Utilities Enhancement Tool encountered an error while trying to locate all active threads on the specified DB2 subsystem.

User response: Contact IBM Software Support.

ABPB6251E Thread list build failed

Explanation: DB2 Utilities Enhancement Tool successfully located information for all active threads on the DB2 subsystem. However, an error occurred when DB2 Utilities Enhancement Tool was retrieving the list of active threads from the started task address space.

User response: Contact IBM Software Support.

ABPB6252E An error occurred related to issuing the DB2 CANCEL THREAD command

Explanation: An error occurred that is related to DB2 Utilities Enhancement Tool issuing the DB2 CANCEL THREADS command. This error could have occurred immediately before or after the command was issued or while the command was being issued. If the error occurred after the command was issued, it is possible that the thread will still be canceled.

User response: Check the started task log to determine whether the DB2 CANCEL THREADS command was actually issued to the DB2 subsystem. If you need assistance, contact IBM Software Support.

ABPB6253E A DB2 error occurred as a result of an IFI call to cancel a thread, RC=*return_code*

Explanation: A DB2 error occurred as a result of an instrumentation facility interface (IFI) call to cancel threads. Usually, this message is issued when a thread ends normally (not because of a Cancel command) between the time that DB2 Utilities Enhancement Tool selects it for cancelation and actually issues the Cancel command. In this case, the message does not indicate an actual error. However, the message might be issued in some other situations. The text of the DB2 error message that was issued is contained in message ABPS0211I, which follows this message. Message

ABPS0211I might be issued multiple times if multiple DB2 errors occurred.

User response: Review the message ABPS0211I for the CANCEL_THREADS request to determine the cause of the error. If the thread stopped normally (not because of a Cancel command), no action is required. If you need assistance, contact IBM Software Support.

ABPB6254E Failed to release storage after retrieval of messages from the started task, RC=*return_code*

Explanation: An error occurred while DB2 Utilities Enhancement Tool was attempting to release storage that was used for cancelation messages from the DB2 Utilities Enhancement Tool started task.

User response: For more information, check the started task logs. If you need assistance, contact IBM Software Support.

ABPB6255E An error occurred while checking the status of a canceled thread, RC=*return_code*

Explanation: An error occurred while DB2 Utilities Enhancement Tool was checking the status of a thread that was canceled.

User response: Look up the return code in the DB2 Utilities Enhancement Tool documentation. If you need assistance, contact IBM Software Support.

ABPB6256E An error occurred while retrieving a result set from the started task, RC=*return_code*

Explanation: An error occurred during the retrieval of information from the DB2 Utilities Enhancement Tool started task. This information could be a list of active threads, a set of messages that were returned from a cancel command, details about a thread, or the objects referenced by a thread.

User response: For more information, look up the return code in the *DB2 Utilities Enhancement Tool for z/OS User's Guide*. If you need assistance, contact IBM Software Support.

ABPB6257E An error occurred related to issuing the command to cancel a thread with escalation

Explanation: An error occurred that is related to DB2 Utilities Enhancement Tool issuing the escalated cancel command. This error could have occurred immediately before or after the command was issued or while the command was being issued. If the error occurred after the command was issued, it is possible that the thread will still be canceled.

User response: Check the started task log to determine whether the escalated cancel command was actually issued. If you need assistance, contact IBM Software Support.

ABPB6258I PLAN=*db2_plan_name*
CONN=*connection_id*
CORR=*correlation_id*
AUTH=*authorization_id*
OAUTH=*original_authorization_id*
TOKN=*thread_token*

Explanation: An escalated cancel command has been issued for the thread that is identified by the attributes in the message text. These attributes are plan name, connection ID, correlation ID, authorization ID, original authorization ID, and thread token. Additional information is provided in the message ABPB6259I.

User response: No action is required.

ABPB6259I JOBN=*job_name* **ASID=***session_asid*
PROGNAME=*program_name*
COLLID=*package_collection_id*

Explanation: This message accompanies the message ABPB6450I. It provides the following additional information about the canceled thread: the job name, the address space identifier (ASID), the program name (DBRM name), and the collection ID of the package under which the thread is running (if applicable).

User response: No action is required.

ABPB6260E An error occurred while processing the command to cancel a thread with escalation, RC=*return_code*

Explanation: An error occurred while the DB2 Utilities Enhancement Tool started task was processing the command to perform an escalated cancelation. Detailed information regarding the cause of the error is in the message that follows this one.

User response: To determine the cause of the error, see the message that follows this one. If you need assistance, contact IBM Software Support.

ABPB6261I Thread cancel with escalation issued:

Explanation: An escalated cancel command has been issued for a thread. The attributes of this thread are provided in the messages ABPB6258I and ABPB6259I, which follow this message.

User response: No action is required.

ABPB6262E Failed to retrieve DB2 information from the started task for DB2SSID=*db2_subsystem_id*

Explanation: DB2 Utilities Enhancement Tool encountered an error while trying to retrieve

information regarding the specified DB2 subsystem.

User response: For more information, check the started task logs. If you need assistance, contact IBM Software Support.

ABPB6263E Failed to release storage after retrieving DB2 information from the started task.

Explanation: An error occurred while DB2 Utilities Enhancement Tool was attempting to release storage that was used for DB2 information retrieved from the DB2 Utilities Enhancement Tool started task.

User response: For more information, check the started task logs. If you need assistance, contact IBM Software Support.

ABPB6264W Unable to retrieve DB2 information from the started task for DB2SSID=*db2_subsystem_id*

Explanation: The DB2 Utilities Enhancement Tool started task could not retrieve thread information for the specified subsystem from DB2 for batch thread-cancelation processing.

User response: Resubmit the batch job that contains the cancel request.

ABPB6265W Unable to locate active threads on the DB2 subsystem, DB2SSID=*db2_subsystem_id*

Explanation: DB2 Utilities Enhancement Tool could not locate the active threads to cancel on the specified DB2 subsystem because DB2 returned no thread data for that subsystem.

User response: Resubmit the batch job that contains the cancel request.

ABPB6410I Thread list build was successful: Active thread count=*thread_count*

Explanation: DB2 Utilities Enhancement Tool finished locating and retrieving information for all active threads on the DB2 subsystem. This message indicates the number of active threads detected.

User response: No action is required.

ABPB6411I Thread filtering applied: Threads of cancel interest count=*thread_count*

Explanation: The thread-filtering parameters for the CANCEL_THREADS request were applied to all of the active threads on the DB2 subsystem. The number of threads that matched all of the thread-filtering criteria is indicated. These threads are eligible for cancelation processing.

User response: No action is required.

ABPB6412W No active threads are eligible for thread cancelation.

Explanation: None of the active threads on the DB2 subsystem are eligible for thread cancelation. They are not eligible because they either did not match the thread-filtering criteria that were specified or they were for DB2 Utilities Enhancement Tool started task configurations. (Started task threads are not canceled.) As a result, the CANCEL_THREADS request ended with the return code RC=04, and no threads were canceled.

User response: No action is required.

ABPB6414I No active threads matched thread filters that specify DB2 objects

Explanation: Thread-filtering parameters that specify DB2 objects were applied to all active threads on the DB2 subsystem for the CANCEL_THREADS request. No active threads matched all of the object filtering criteria. As a result, the CANCEL_THREADS request ends with the return code RC=04.

User response: No action is required.

ABPB6430I Thread object filtering applied. Cancel thread interest count=*thread_count*

Explanation: The thread filtering parameters for DB2 objects that were specified in the CANCEL_THREADS request were applied to all active threads. The number of threads that matched the filtering criteria is indicated by the thread interest count. This count indicates the number of threads that are eligible for cancelation processing.

User response: No action is required.

ABPB6450I Thread cancel issued:
PLAN=*db2_plan_name*
CONN=*connection_id*
CORR=*correlation_id*
AUTH=*authorization_id*
OAUTH=*original_authorization_id*
TOKN=*thread_token*

Explanation: A CANCEL THREAD command has been issued for a thread. The following attributes of the thread are displayed in the message text: the plan name, connection ID, correlation ID, authorization ID, original authorization ID, and thread token value. Additional information is provided in the message ABPB6451I.

User response: No action is required.

ABPB6451I **JOBN=***job_name* **ASID=***session_asid* **PROGNAME=***program_name* **COLLID=***package_collection_id*

Explanation: This message accompanies the message ABPB6450I. It provides the following additional information about the canceled thread: the job name, the address space identifier (ASID), the program name (DBRM name), and the collection ID of the package under which the thread is running (if applicable).

User response: No action is required.

ABPB6461I **Thread has terminated: PLAN=***db2_plan_name* **CONN=***connection_id* **CORR=***correlation_id* **AUTH=***authorization_id* **OAUTH=***original_authorization_id* **TOKN=***thread_token*

Explanation: DB2 Utilities Enhancement Tool checks threads that were canceled to determine if they have actually terminated in DB2. This message is issued when this checking detects that a thread has terminated. The following attributes of the terminated thread are provided in the message text: the plan name, connection ID, correlation ID, authorization ID, original authorization ID, and thread token value.

User response: No action is required.

ABPB6462I **Thread *still active*: PLAN=***db2_plan_name* **CONN=***connection_id* **CORR=***correlation_id* **AUTH=***authorization_id* **OAUTH=***original_authorization_id* **TOKN=***thread_token*

Explanation: DB2 Utilities Enhancement Tool checks threads that were canceled to determine if they have actually terminated in DB2. This message is issued when this checking detects that a canceled thread is still active. The following attributes of the active thread are displayed: the plan name, connection ID, correlation ID, authorization ID, original authorization ID, and thread token value.

User response: No action is required.

ABPB6470I **Thread status checking for canceled threads started. Retry count=***retry_count* **Retry interval=***retry_interval*

Explanation: DB2 Utilities Enhancement Tool has started checking the threads that were canceled to determine if they have actually terminated in DB2. The retry count is the maximum number of times that termination checking will be retried. The retry interval is the number of seconds between retry attempts.

User response: No action is required.

ABPB6471I **Thread status checking ended; all canceled threads have terminated**

Explanation: All of the threads that were canceled have actually terminated. DB2 Utilities Enhancement Tool will stop checking the threads to determine their termination status.

User response: No action is required.

ABPB6472E **Thread status checking ended. Canceled threads *still active* count=***thread_count***

Explanation: DB2 Utilities Enhancement Tool stopped checking the threads that were canceled to determine whether they actually terminated because the checking retry count was reached. At least one canceled thread remains active.

User response: Try to determine why the active thread or threads were not terminated. You can view thread information in the DB2 Utilities Enhancement Tool ISPF interface and the output for the CANCEL_THREADS request. If you need assistance, contact IBM Software Support.

ABPB6473I **Thread status checking is being retried. Threads still active=***thread_count* **Retries remaining=***retry_attempts*

Explanation: Thread status checking identified threads that were canceled but have not yet terminated. The number of canceled threads that are still active is indicated. Also, the number of times that thread status checking can be retried before the maximum retry count is reached is indicated.

User response: No action is required.

ABPB6543W **Failed to determine the objects referenced by thread** *thread_token* **for report** *rept_report_dd_name***, RC=***return_code*

Explanation: An error occurred while DB2 Utilities Enhancement Tool was identifying the set of DB2 objects that are referenced by the thread that has the specified thread token value.

User response: For more information, look up the return code in the DB2 Utilities Enhancement Tool user guide. If you need assistance, contact IBM Software Support.

ABPB6544W **Failed to get thread detail data for thread, TTOKEN=***thread_token*

Explanation: DB2 Utilities Enhancement Tool could not obtain detailed information for the specified thread. As a result, the batch reports that were generated for the current cancel request might not contain some data for the thread.

User response: No action is required.

ABPB6545I SEQNO CAN TTOKEN PLANNAME AUTHID UR_STRT_DT UR_STRT_TIME UR_STATE UR_ELAP_TIME UR_LOG_SRBA/ LOG_PAGES LOG_RECS

Explanation: This message is the first line of the column header in the All Active Threads Unit of Recovery Report. This message appears in the report without the message identifier.

User response: No action is required.

ABPB6546I UR_LOG_ERBA

Explanation: This message is the second line of the column header in the All Active Threads Unit of Recovery Report. This message appears in the report without the message identifier.

User response: No action is required.

ABPB6547I ----- ---

Explanation: This message is the last line of the column header in the All Active Threads Unit of Recovery Report. This message appears in the report without the message identifier.

User response: No action is required.

ABPB6548I data_entry

Explanation: This message is a data entry in the All Active Threads Unit of Recovery Report. It is issued when Unit of Recovery information is available for the specified thread. (Otherwise, ABPB6533I is issued.) This message corresponds to the column header message ABPB6545I and appears in the report without the message identifier.

User response: No action is required.

ABPB6549I URLERBA

Explanation: This message is the second line of a data entry in the All Active Threads Unit of Recovery Report. It is issued when Unit of Recovery information is available for the specified thread. (Otherwise, ABPB6533I is issued.) This message corresponds to the column header message ABPB6546I and appears in the report without the message identifier.

User response: No action is required.

ABPB6550I SEQNO TTOKEN PLANNAME AUTHID UR_STRT_DT UR_STRT_TIME UR_STATE UR_ELAP_TIME UR_LOG_SRBA/ LOG_PAGES LOG_RECS

Explanation: This message is the first line of the column header in the Threads Canceled Unit of Recovery Report. This message appears in the report without the message identifier.

User response: No action is required.

ABPB6551I UR_LOG_ERBA

Explanation: This message is the second line of the column header in the Threads Canceled Unit of Recovery Report. This message appears in the report without the message identifier.

User response: No action is required.

ABPB6552I ----- ---

Explanation: This message is the last line of the column header in the Threads Canceled Unit of Recovery Report. This message appears in the report without the message identifier.

User response: No action is required.

ABPB6553I data_entry

Explanation: This message is a data entry in the Threads Canceled Unit of Recovery Report. It is issued when Unit of Recovery information is available for the specified thread. (Otherwise, ABPB6516I is issued.) This message corresponds to the column header message ABPB6550I and appears in the report without the message identifier.

User response: No action is required.

ABPB6554I data_entry

Explanation: This message is the second line of a data entry in the Threads Canceled Unit of Recovery Report. It is issued when Unit of Recovery information is available for the specified thread. (Otherwise, ABPB6516I is issued.) This message corresponds to the column header message ABPB6551I and appears in the report without the message identifier.

User response: No action is required.

ABPB6700I DB2 Utilities Enhancement Tool cancel execution ended. Highest RC=*return_code*

Explanation: The thread-cancelation job ended. This message provides the highest return code from all of the CANCEL_THREADS requests in the job.

User response: If the return code is greater than 4, an error might have occurred. For more information, look up the return code in the DB2 Utilities Enhancement Tool documentation. You can also view the messages in the job output to determine why the return code was issued.

ABPB6800I MODULE LEVEL DATE TIME EPA RREPA CC F1 F2 F3 SEQ

Explanation: This message displays the fields in the Module Entry Point List (MEPL) control block.

User response: No action is required.

ABPB6801I *module_name maintenance_level assembly_date assembly_time entry_point_address rr_entry_point_address component_code flag_byte_1 flag_byte_2 flag_byte_3 sequence_number*

Explanation: This message displays the data in the fields of the Module Entry Point List (MEPL) control block.

User response: No action is required.

ABPB6998I *batch_input_parm*

Explanation: This message displays the value for a parameter that was specified for the CANCEL_THREADS request. If the parameter value is long and requires additional lines, it is continued in the message ABPB6999I.

User response: No action is required.

ABPB6999I *batch_input_parm*

Explanation: This message follows ABPB6998I if the parameter value that is being reported for the CANCEL_THREADS request is long and cannot fit in its entirety in the message ABPB6998I. This message contains the continued value.

User response: No action is required.

ABPG8000S Internal error in API <*api_context***>, RC=<***api_return_code***>, RSN=<***api_reason_code***>.**

Explanation: An error occurred in the DB2 Utilities Enhancement Tool internal application programming interface (API).

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPG8001S Storage release failed. Module=<*module_name***>, storage area=<***storage_area_name***>, RC=<***return_code***>.**

Explanation: The specified module failed while attempting to free to the specified storage area. The message ABPG8002S, which accompanies this message, contains additional information about the storage area.

User response: Contact IBM Software Support. Provide the support representative with the complete text of this message and message ABPG8002S.

ABPG8002S Address=*storage_area_address***, Length=***storage_area_length***, SP=***storage_subpool***, KEY=***storage_key***

Explanation: This message accompanies message ABPG8001S, which indicates a failure to release storage. This message provides additional details about the storage that could not be released.

User response: Contact IBM Software Support. Provide Support with the complete text of this message and message ABPG8001S.

ABPG8003E Storage obtain failed. Module=<*module_name***>, storage area=<***storage_area_name***>, RC=<***return_code***>.**

Explanation: The specified module failed while attempting to obtain the specified storage area. The message ABPG8004E usually accompanies this message and contains additional information about the storage area.

User response: Increase the region size available to the DB2 Utilities Enhancement Tool program and run the product again. If the problem persists, contact IBM Software Support. Provide Support with the complete text of this message and message ABPG8004E.

ABPG8004E Length=*storage_area_length***, SP=***storage_subpool***, KEY=***storage_key***

Explanation: This message accompanies the message ABPG8003E, which indicates a failure to obtain storage. This message provides additional details about the storage that could not be obtained.

User response: Increase the region size available to the DB2 Utilities Enhancement Tool program and run the product again. If the problem persists, contact IBM Software Support. Provide Support with the complete text of this message and message ABPG8003E.

ABPG8005E Unable to open file. DD name=*dd_name*

Explanation: The file that was allocated by the specified data definition (DD) could not be opened.

User response: Check the JCL to ensure that the correct DD name was provided and that the data set was allocated using the correct file type.

ABPG8006E Unable to dynamically allocate data set. DD name=*dd_name*

Explanation: The specified data definition (DD) was not able to dynamically allocate a data set that was needed.

User response: Contact IBM Software Support.

ABPG8007E Unable to close file. DD name=*dd_name*

Explanation: The file that was allocated by the specified data definition (DD) could not be closed.

User response: If this problem persists, contact IBM Software Support.

ABPG8008I System=*system_name***, Job=***job_name***, Job Id=***job_id***, Step=***step_name***, Program=***program_name***, User=***user_id*

Explanation: This message displays information about the current job step.

User response: No action is required.

ABPG8009E The operating system or hardware do not meet minimum requirements.

Explanation: See the Product Program Directory for the minimum operating system level and hardware requirements.

User response: No action is required.

ABPG8010I CPU=<*cpu_type***>, <***cpu_model***>, <***cpu_manufacturer***>. OS=<***os_name***>, <***os_release***>, <***os_version***>.**

Explanation: This message displays information about the CPU and the operating system.

User response: No action is required.

ABPM9600E An invalid function was supplied to utility.

Explanation: An invalid function was specified in the ABPMAINT job for the DB2 Utilities Enhancement Tool maintenance utility.

User response: In the PARM statement of the ABPMAINT job, specify a valid function (for example, TERM_UTILITY). See the user's guide for the functions that are valid for the ABPMAINT utility.

ABPM9601E API Initialization failed

Explanation: The ABPMAINT interface program failed to complete initialization. This failure occurred during the initialization of the internal API.

User response: Contact IBM Software Support.

ABPM9602E Unable to establish session with ABPID: *identifier*

Explanation: The DB2 Utilities Enhancement Tool maintenance utility could not establish a session with the specified started task configuration.

User response: Check that the configuration ID parameter value that is specified in the maintenance utility job (ABPMAINT) is a valid configuration ID.

ABPM9603E Unable to connect to DB2 system: *db2_ssid*

Explanation: The DB2 Utilities Enhancement Tool maintenance utility could not connect to the specified DB2 subsystem.

User response: Ensure that the DB2 SSID parameter value that is specified in the maintenance utility job (ABPMAINT) specifies a valid DB2 subsystem ID.

ABPM9604I Worklist maintenance successful for utility id: *db2_utility_id***, function:** *maint_utility_function*

Explanation: The DB2 Utilities Enhancement Tool maintenance utility successfully performed the specified function for the specified DB2 utility ID in the worklist tables.

User response: No action is required.

ABPM9605W No worklist data found for UTILID: *db2_utility_id***, function:** *maint_utility_function*

Explanation: The DB2 Utilities Enhancement Tool maintenance utility found no worklist data for the specified DB2 utility ID. The specified function could not be performed.

User response: No action is required.

ABPM9606E Error while accessing worklist data for utility ID: *db2_utility_id***, function:** *MAINT_function*

Explanation: The DB2 Utilities Enhancement Tool maintenance utility (ABPMAINT) encountered an error while attempting to access the worklist data that is associated with the specified DB2 utility ID. The specified maintenance utility function could not be performed.

User response: In the ABPMAINT job, check that the PARM statement specifies valid values for the DB2 SSID and utility ID parameters. Also check the messages in the started task SYSPRINT log for related SQL errors.

ABPM9607E Session creation failed RC=<*return_code***>, RSN=<***reason_code***>, reason=<***description***>.**

Explanation: The DB2 Utilities Enhancement Tool maintenance utility (ABPMAINT) failed to complete initialization. The failure occurred during the creation of a session for ABPMAINT.

User response: To determine the cause of the failure, review the reason description in this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPM9608E Session has been terminated by the server.

Explanation: The utility did not complete because the session was terminated by the server.

User response: Check with the system administrator to determine the reason for the termination of the maintenance utility program.

ABPP9800E <!ATTLIST *attribute_name*> attribute '*attribute_value*' has an invalid enumeration value list

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an !ATTLIST declaration, it found an error in the enumeration value list that defines the valid values for an attribute.

User response: Correct the enumeration list and rerun.

ABPP9801E <!ATTLIST *attribute_name*> attribute '*attribute_value*' missing enumeration value

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an !ATTLIST declaration, it encountered an invalid enumeration token. Enumeration tokens must be valid XML names.

User response: Correct the enumeration list and rerun.

ABPP9802E <!ATTLIST *attribute_name*> attribute '*attribute_value*' no closing quote for default value

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an !ATTLIST declaration, it found a default value of type *string*, but it did not have a closing quotation mark.

User response: Correct the string definition and rerun.

ABPP9803E <!ATTLIST *attribute_name*> attribute '*attribute_value*' enumerated type list missing '('

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an !ATTLIST declaration, it encountered an enumeration or NOTATION list, as expected, but that list did not have an opening left parenthesis.

User response: Correct the enumeration list and rerun.

ABPP9804E <!ATTLIST *attribute_name*> attribute '*attribute_value*' expected quoted default value

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an !ATTLIST declaration, it did not find the default value specification, as expected.

User response: Supply a default value for the !ATTLIST declaration and rerun.

ABPP9805E getAttribute(*missing_attribute_name*) error: attribute not defined

Explanation: A request was made to retrieve the value of an attribute for a given XML element, but the attribute was not defined.

User response: Verify that the attribute exists before requesting its value, or add the attribute to the XML document.

ABPP9806E '<![IGNORE[' not terminated by matching ']]>'

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing an <![IGNORE[...]]> conditional section, it did not find the required closing character sequence.

User response: Correct the conditional sequence and rerun.

ABPP9807E '<![INCLUDE[' not terminated by matching ']]>'

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing an <![INCLUDE[...]]> conditional section, it did not find the required closing character sequence.

User response: Correct the conditional sequence and rerun.

ABPP9808E Unable to open default input file

Explanation: The XML parser was not able to open the top-level default input file.

User response: Make sure that the file exists and the correct name is being passed to the XML parser.

ABPP9809E Unable to open DOCTYPE file '*DOCTYPE_file_name*'

Explanation: A !DOCTYPE declaration was specified, but the DOCTYPE file name could not be read.

User response: Make sure that the DOCTYPE file exists and that the correct file name is specified in the !DOCTYPE declaration.

ABPP9810E Unable to open external ENTITY file '*ENTITY_file_name*'

Explanation: An external entity file was defined, but it could not be read to resolve the entity reference.

User response: Make sure that the external entity file exists and that the correct file name is specified in the !ENTITY declaration.

ABPP9811E element <*element_name*> ended by </*element_name_1*>

Explanation: An incorrectly nested element definition was found. The tag defining the beginning of an element did not match the closing tag.

User response: Correct the nesting structure of the element definition and rerun.

ABPP9812E Closing tag </*element_name* missing '>' character

Explanation: When the XML parser was parsing the end tag for an element, it did not find the required closing '>' character.

User response: Correct the end tag and rerun.

ABPP9813E <!ELEMENT *element_name*> is already declared

Explanation: Only one !ELEMENT declaration can be supplied for a given element tag.

User response: Remove the duplicate !ELEMENT declaration and rerun.

ABPP9814E <!ELEMENT *element_name*> expecting subelement name.

Explanation: When the XML parser was parsing a mixed-content specification of an !ATTLIST declaration, it found an error in the list of allowable subelements.

User response: Correct the subelement list and rerun.

ABPP9815E ENTITY &*entity_name*; not defined

Explanation: An entity reference was found for which no declaration exists.

User response: Check the spelling of the entity reference name, or add the entity definition for the name and rerun.

ABPP9816E End-of-data encountered while parsing attribute value string

Explanation: When the XML parser was parsing an attribute value string, it found no closing quotation mark before the end of the file, as required.

User response: Correct the attribute value string and rerun.

ABPP9817E End-of-data encountered in a CDATA section

Explanation: When the XML parser was parsing a <![CDATA[...]]> section, it found no ']]>' characters. These characters are required to close the section before the end of the file.

User response: Correct the CDATA section and rerun.

ABPP9818E End-of-data encountered in a comment

Explanation: When the XML parser was parsing an XML comment, it found no '-->' characters. These characters are required to close the comment before the end of the file.

User response: Correct the comment and rerun.

ABPP9819E End-of-data encountered inside a declaration

Explanation: When the XML parser was parsing an XML declaration, it found no '>' character. This character is required to close the declaration before the end of the file.

User response: Correct the declaration and rerun.

ABPP9820E End-of-data encountered in DOCTYPE declaration

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing a !DOCTYPE declaration, it reached the end of the file before the declaration was complete.

User response: Correct the !DOCTYPE declaration and rerun.

ABPP9821E End-of-data encountered while parsing element attributes

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing the attribute list for an element, it reached the end of the file before the attribute list was complete.

User response: Correct the element attribute list and rerun.

ABPP9822E End-of-data encountered inside an <!ELEMENT ...> declaration

Explanation: When the XML parser was parsing an !ELEMENT declaration, it reached the end of the file before the declaration was complete.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9823E End-of-data encountered in ENTITY definition

Explanation: When the XML parser was parsing an !ENTITY declaration, it reached the end of the file before the declaration was complete.

User response: Correct the !ENTITY declaration and rerun.

ABPP9824E End-of-data encountered in processing instruction

Explanation: When the XML parser was parsing an XML processing instruction, it reached the end of the file before the processing instruction was complete.

User response: Correct the processing instruction and rerun.

ABPP9825E Invalid <!ATTLIST *attribute_name*> attribute name

Explanation: A syntax error was detected while the DB2 Utilities Enhancement Tool XML parser was parsing an XML !ATTLIST declaration.

User response: Correct the !ATTLIST declaration and rerun.

ABPP9826E Invalid DOCTYPE name

Explanation: When the XML parser was parsing an XML !DOCTYPE declaration, it found no valid element name.

User response: Correct the !DOCTYPE declaration and rerun.

ABPP9827E Invalid !ELEMENT name

Explanation: When theDB2 Utilities Enhancement Tool XML parser was parsing an XML !ELEMENT declaration, it found no valid element name.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9828E Invalid element tag

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing an XML statement, it did not find an expected element tag.

User response: Correct the error and rerun.

ABPP9829E Invalid name in ENTITY definition

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing an ENTITY definition, it found no valid entity name.

User response: Correct the ENTITY declaration and rerun.

ABPP9830E Invalid ENTITY reference

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing an entity reference, it found no semicolon to terminate the entity reference. Check if an ampersand is incorrectly specified in a string as '&'. An ampersand should be specified as &.

User response: Correct the entity reference and rerun.

ABPP9831E Invalid value in ENTITY definition: *value*

Explanation: A syntax error was encountered while the DB2 Utilities Enhancement Tool XML parser was parsing an XML ENTITY definition.

User response: Correct the ENTITY definition and rerun.

ABPP9832E typespec for <!ELEMENT *element_name*> not correctly ended

Explanation: A syntax error was detected while DB2 Utilities Enhancement Tool was processing the typespec parameter of an !ELEMENT declaration.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9833E '<' character not legal in attribute value string

Explanation: The replacement text of any entity referred to directly or indirectly in an attribute value must not contain a '<' character.

User response: Correct the attribute value and rerun.

ABPP9834E No attributes defined for non-element node types

Explanation: An attempt was made to request an attribute for an XML element type that does not have attributes.

User response: Make sure that you have an XML element object before you request an attribute value.

ABPP9835E Attribute name not found

Explanation: The XML parser was expecting an attribute name, but no valid attribute name was found.

User response: Correct the XML statement and rerun.

ABPP9836E No closing ']' for DOCTYPE internal subset definition

Explanation: When the DB2 Utilities Enhancement Tool XML parser was parsing an entity definition list in an XML !DOCTYPE declaration, it found no closing ']'

character. The closing character is required.

User response: Correct the !DOCTYPE declaration and rerun.

ABPP9837E No closing '>' for ENTITY definition: *entity_name*

Explanation: No closing '>' character was found to indicate the end of an ENTITY definition.

User response: Correct the ENTITY definition and rerun.

ABPP9838E No closing tag for <*element_name***>**

Explanation: The XML parser was expecting to find a closing tag for the element but did not find it.

User response: Correct the XML element declaration and rerun.

ABPP9839E No '=' following attribute name '*attribute_name***'**

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an attribute definition, it expected an '=' sign but found something else. The XML language does not allow spaces before or after the '=' sign in an attribute definition. If these spaces exist, remove them.

User response: Correct the attribute definition and rerun.

ABPP9840E Tag does not follow '<'

Explanation: An XML element tag must immediately follow the opening '<' character of an element definition. The XML parser found a white space character following the '<' instead.

User response: Fix the element definition and rerun.

ABPP9841E Tag does not follow '</'

Explanation: An XML element tag must immediately follow the closing '</' character of an element definition. The XML parser found a white space character following the '</' instead.

User response: Fix the element definition and rerun.

ABPP9842E No value found for attribute '*attribute_name***'**

Explanation: While the DB2 Utilities Enhancement Tool XML parser was parsing an attribute definition, it expected a value to follow the '=' character but found no valid value at that location. The XML language does not allow blanks before or after the '=' character in an attribute definition. If these blanks exist, remove them.

User response: Correct the attribute definition and rerun.

ABPP9843E parameter %*parameter_name***; is not defined**

Explanation: An undefined parameter reference was found.

User response: Check the spelling of the parameter name, or add a definition for the parameter and rerun.

ABPP9844E Unexpected character following DOCTYPE SYSTEM name

Explanation: The XML parser expected a '>' character to close a !DOCTYPE declaration but found something else.

User response: Correct the !DOCTYPE declaration and rerun.

ABPP9845E Unexpected character in <!ELEMENT *element_name***> children**

Explanation: A syntax error was detected while the XML parser was parsing the list of child elements allowed for an !ELEMENT declaration.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9846E Unexpected characters following <!ELEMENT *element_name* **(#PCDATA**

Explanation: The XML parser expected to find a closing ')' character for the #PCDATA token but found something else.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9847E Unexpected characters in <!ELEMENT *element_name***> contentspec**

Explanation: The XML parser detected an unexpected character following the #PCDATA portion of an !ELEMENT declaration.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9848E Unexpected contentspec <!ELEMENT *element_name***> declaration**

Explanation: A syntax error was detected in the contentspec portion of an !ELEMENT declaration.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9849E Unexpected !DOCTYPE option: *option*

Explanation: The XML parser detected an invalid option in a !DOCTYPE declaration.

User response: Correct the !DOCTYPE declaration and rerun.

ABPP9850E Unexpected !DOCTYPE SYSTEM value

Explanation: The XML parser detected an invalid value in the SYSTEM portion of a !DOCTYPE declaration.

User response: Correct the !DOCTYPE declaration and rerun.

ABPP9851E Unexpected !ENTITY SYSTEM value

Explanation: The XML parser detected an invalid value in the SYSTEM portion of a !ENTITY declaration.

User response: Correct the !ENTITY declaration and rerun.

ABPP9852E Unexpected text in <!ELEMENT *element_name***>**

Explanation: While the XML parser was parsing an !ELEMENT declaration, it expected to find a closing '>' character but found something else.

User response: Correct the !ELEMENT declaration and rerun.

ABPP9853E Quotation delimiters do not match for attribute value *attribute_value*

Explanation: The delimiter characters around the specified attribute value in the DSNUTILB intercept policy do not match. The delimiter characters must both be either double quotation marks or single quotation marks.

User response: Correct the delimiters that enclose the specified attribute value so that they match. Use either double quotation marks or single quotation marks. Then rerun the utility.

ABPP9854W USE_RULESET element in POLICY references an undefined ruleset. Name: *ruleset_name***.**

Explanation: In the DSNUTILB intercept policy, a <USE_RULESET> element in the <POLICY> section references a ruleset name that has not been defined by a <RULESET> element.

User response: Ensure that the ruleset name that is specified by the <USE_RULESET> element matches a ruleset name that is defined by a <RULESET> element in the same policy. You can either correct the ruleset name that is specified by the <USE_RULESET> element or change the ruleset name that is defined by the <RULESET> element (if that ruleset is not referenced by other <USE_RULESET> elements in the policy).

ABPP9855W VRUPDATE element omitted after ACTION=VRUPDATE for DB2SYSTEM *db2_ssid***.**

Explanation: In the DSNUTILB policy, an ACTION=VRUPDATE attribute on the DB2SYSTEM element requires a VRUPDATE child element to be included under the DB2SYSTEM element.

User response: Ensure that the VRUPDATE element is included and that the VRUPDATE element has a DSN attribute that specifies the VR UPDATE job JCL.

ABPP9856W Usage of RULE SYNONYM has been deprecated.

Explanation: In the DSNUTILB policy, a RULE SYNONYM= was encountered. The usage of RULE element SYNONYM has been deprecated.

User response: No action is required.

ABPP9857E Invalid characters encountered in PART specification.

Explanation: The XML parser detected an invalid character in the PART specification.

User response: Correct the PART specification and rerun.

ABPP9858E USE_PRACTICE in POLICY references an undefined practice. PRACTICE =*practice_name***.**

Explanation: In the DSNUTILB policy, a USE_PRACTICE element in the POLICY section references a PRACTICE name that has not been defined by a PRACTICE element.

User response: Ensure that the practice name that is specified by the USE_PRACTICE element matches a practice name that is defined by a practice element in the same policy. You can either correct the practice name that is specified by the USE_practice element or change the practice name that is defined by the practice element (if that practice is not referenced by other USE_PRACTICE elements in the policy.

ABPP9859E A duplicate practice name was specified in the policy. PRACTICE=<*practice_name***>.**

Explanation: The DSNUTILB policy defined two PRACTICE elements with the same value specified for the NAME attribute. When PRACTICE elements with duplicate names are found in the policy, the PRACTICE

that is coded first in the policy is used by the utility monitor.

User response: Ensure that all practice names are unique.

ABPP9860E A duplicate utility name was specified in a practice. UTILNAME =*utility_name*

Explanation: The DSNUTILB policy defined two UTILITY elements with the same NAME under a practice.

User response: Ensure that all utility names are unique within a PRACTICE specification.

ABPP9861E The length of attribute is greater than 1024 characters: ATTRIBUTE=*attribute_name***.**

Explanation: Attributes VALUE and SUBSTITUTE of the SYNTAX policy element are each restricted to 1024 characters.

User response: Correct the specified attribute.

ABPP9862E Attribute *attribute_name_1* **is incompatible with attribute** *attribute_name_2***.**

Explanation: The two specified attributes are mutually exclusive and can not be used together.

User response: Correct the attribute specifications.

ABPP9863E Multiple <USE_PRACTICE> elements were specified within one <DB2SYSTEM> element, DB2 SSID: <*db2_ssid***>.**

Explanation: In the DSNUTILB policy, multiple <USE_PRACTICE> elements were specified within one <DB2SYSTEM> element. Each <DB2SYSTEM> element can contain only one <USE_PRACTICE> element.

User response: Make sure that all <DB2SYSTEM> elements contain only one <USE_PRACTICE> element.

ABPP9864E <SUBSTITUTE> or <FAIL> attributes must be specified for <VALUE> attribute in <SYNTAX> element.

Explanation: In the DSNUTILB intercept policy, the <VALUE> attribute in the <SYNTAX> element was specified without the required <SUBSTITUTE> or <FAIL> attribute.

User response: Make sure that the <VALUE> attribute in the <SYNTAX> element was specified with the <SUBSTITUTE> or <FAIL> attribute.

ABPP9876E KEYWORD1 <*keyword***> and KEYWORD2 <***keyword***> are mutually exclusive keywords.**

Explanation: The specified keywords are mutually exclusive. You can specify one or the other, but not both.

User response: Remove one of the keywords and resubmit the job.

ABPP9878E The keyword <*keyword***> is not valid as used.**

Explanation: The specified keyword is not valid in the context in which it is used.

User response: Correct the load statement and resubmit the job.

ABPP9879E Validation error: ID '*enum_value***' not found for IDREF reference**

Explanation: An attribute was declared to be an IDREF, but the attribute value was not used as an ID within the XML document.

User response: Check the spelling of the IDREF value, or add a corresponding ID attribute that uses the IDREF value.

ABPP9880E Validation error: default '*enum_value***' for attribute '***attribute_name***' not a member of enumerated type**

Explanation: The default value that is specified for an attribute in an !ATTLIST declaration of the Document Type Definition is not a valid value for the attribute.

User response: Correct the !ATTLIST declaration so that the default value is one of the values in the enumerated list of valid attribute values, and then rerun.

ABPP9881E Validation error: <!ELEMENT ...*element_name***> attribute value** *attribute_name***='***enum_value***' is not a member of the enumerated type.**

Explanation: The value that is specified for an attribute is not one of the valid values that is defined for the attribute in the Document Type Definition. When the specified value is NULL or blanks, the default value is used.

User response: Correct the attribute value and rerun the job.

ABPP9882E Validation error: attributes declared ID must be #REQUIRED or #IMPLIED

Explanation: An ID attribute must have a declared default of #IMPLIED or #REQUIRED.

User response: Correct the default value for the ID attribute and rerun.

ABPP9883E Validation error: duplicate ID *ID_name*='*value*'

Explanation: A name must not appear more than once in an XML document as an ID value. That is, ID values must uniquely identify elements.

User response: Eliminate the duplicate ID and rerun.

ABPP9884E Validation error: <!ELEMENT *element_name* **EMPTY> cannot have subelement** *subelement_name***.**

Explanation: The Document Type Definition (DTD) does not list the specified subelement as one that is valid for the element.

User response: Correct the element definition to eliminate the invalid subelement and rerun.

ABPP9885E Validation error: <!ELEMENT *element_name* **EMPTY> cannot contain text**

Explanation: An element that is declared to be EMPTY in the Document Type Definition cannot contain any content.

User response: Correct the element definition to remove the content and rerun.

ABPP9886E Validation error: <!ELEMENT *element_name*> **invalid attribute** *attribute_name*='*value*'

Explanation: The attribute is not valid for the element according to the Document Type Definition.

User response: Correct the element definition to remove the invalid attribute and rerun.

ABPP9887E Validation error: <!ELEMENT *element_name*> **attribute** *attribute_name*='*attribute_value*' **not #FIXED default value** '*default_value*'

Explanation: The Document Type Definition specifies that the attribute must have a specific #FIXED value, but the attribute definition specifies a different value.

User response: Correct the attribute to use the #FIXED value and rerun.

ABPP9888E Validation error: <!ELEMENT *element_name*> **unexpected subelement** *subelement_name***.**

Explanation: The specified subelement is not valid in the element according to the Document Type Definition (DTD). This error can occur if the subelement is out-of-order with respect to other subelements, or if it is repeated an incorrect number of times.

User response: Correct the element definition so that it is consistent with the DTD.

ABPP9889E Validation error: <!ELEMENT *element_name*> **missing required attribute** '*required_attribute*'

Explanation: An attribute of the element was declared as #REQUIRED in the Document Type Definition, but the attribute is not defined in the element definition.

User response: Correct the element definition to include the required attribute and rerun.

ABPP9890E Validation error: <!ELEMENT *element_name*> **missing required choice** '*required_choice*'

Explanation: The Document Type Definition specifies that one of the valid choices defined for the specified element must appear at a specific position within the element. However, the value that was found at that position is not one of the valid choices.

User response: Correct the element definition to conform to the Document Type Definition and rerun.

ABPP9891E Validation error: <!ELEMENT *element_name*> **missing required subelement** *subelement_name***.**

Explanation: The Document Type Definition (DTD) indicates that the specified subelement is required for the element. However, the subelement was not found in the element definition.

User response: Correct the element definition to supply the required subelement and rerun.

ABPP9892E Validation error: An element can have only one attribute of type ID

Explanation: An element type must not have more than one ID attribute specified.

User response: Correct the !ELEMENT definition so that it has only one ID attribute and rerun.

ABPP9893E Validation error: Element <*element_name*> has not been declared

Explanation: The element that is being defined is not declared in the Document Type Definition. Therefore, it is not permitted in the document.

User response: Eliminate the invalid element definition and rerun.

ABPP9894E XML Parser Exception: *file_name* **line=***line_number*

Explanation: The XML parser error that was previously reported caused the parser to stop. The error occurred while the parser was processing data from the specified file at the specified line number.

User response: Correct the error and rerun.

ABPP9895E XML Parser Exception occurred while processing line=*line_number*

Explanation: The XML parser error that was previously reported caused the parser to stop. The error occurred while the parser was processing data from the top-level input file at the specified line number.

User response: Correct the error and rerun.

ABPP9896E XML Parser Exception

Explanation: The XML parser error that was previously reported caused the parser to stop. The error could not be attributed to a specific line in an input file.

User response: Correct the previously noted error and rerun.

ABPP9897E Allocation error for policy control block : *enum_value*

Explanation: An allocation error occurred for one of the policy control blocks.

User response: Contact IBM Software Support. Provide Support with the message number and text.

ABPP9898E Value exceeds maximum length for RULE: *rule_name*='*rule_value*'

Explanation: In the DSNUTILB policy, the length of the specified rule value exceeds the maximum allowable length for the rule. This message provides the first 32 bytes of the rule value that is in error.

User response: Correct the specified rule value in the DSNUTILB policy so that it does not exceed the maximum allowable length for the rule. See the *DB2 Utilities Enhancement Tool for z/OS User's Guide* for information about maximum allowable rule lengths. After you make the correction, resubmit the job.

ABPP9899E Policy parser error.

Explanation: The DSNUTILB policy parser encountered an error that caused it to stop.

User response: Look for the messages that follow this one for a more detailed description of the error. If the error is related to a policy syntax error, correct the policy and then resubmit the job.

ABPP9900E Abnormal termination: *file_name* **line=***line_number*

Explanation: The XML parser terminated because of an internal error.

User response: Report the problem to IBM Software Support.

ABPP9901E Error: Input buffer size (*size* bytes) is too small.

Explanation: The input buffer for the XML parser overflowed.

User response: The parser must be rebuilt with a larger input buffer size. Contact IBM Software Support.

ABPP9902E Invalid value specified for IGNOREFIELDS. Valid values and YES and NO.

Explanation: An invalid value was specified for the IGNOREFIELDS option of the LOAD utility INTO TABLE statement.

User response: Correct the LOAD syntax and resubmit the job.

ABPP9910I *parsed_batch_syntax*

Explanation: The DB2 Utilities Enhancement Tool batch interface writes this message to the SPRT0000 output for the thread-cancelation job. This message indicates the parameters or cancel commands that were parsed from the ABPPARMS DD in the job.

User response: No action is required.

ABPP9911W '*parsed_batch_syntax*' can only occur once. Only the last occurrence is used to process the request.

Explanation: The specified parameter occurs more than once in the ABPPARMS DD of the batch thread-cancelation job. DB2 Utilities Enhancement Tool will process the last occurrence of the parameter and ignore all previous occurrences.

User response: No action is required.

ABPP9912E **'*batch_parameter_value*' is not a recognized value**

Explanation: An invalid value was specified for a parameter in the batch thread-cancelation job. The parameter name and value have been written to the SPRT0000 output for the job.

User response: Look up the valid values for this parameter in the product documentation. Correct the parameter value in the ABPPARMS DD of the job and run the job again.

ABPP9913E **Value '*user_specified_value*' must be from *minimum_valid_value* through *maximum_valid_value***

Explanation: The specified value is not within the range of valid values for this batch parameter. The parameter name and value have been written to the SPRT0000 output for the thread-cancelation job.

User response: Look up the valid values for this parameter in the product documentation. Correct the parameter value in the ABPPARMS DD of the job and run the job again.

ABPP9914E ***parameter_value* value can be at most *maximum_length* bytes**

Explanation: The specified batch job parameter value is longer than the maximum length that is allowed for this parameter.

User response: Correct the parameter value in the ABPPARMS DD of the batch thread-cancelation job. Ensure that it does not exceed the maximum length that is specified in this message text. Then run the job again.

ABPP9915E **Expected value '*expected_value*' not found**

Explanation: DB2 Utilities Enhancement Tool expected the specified value to be in the ABPPARMS DD of the batch thread-cancelation job but did not find it there.

User response: Add the specified value to the ABPPARMS DD where appropriate. Then run the job again.

ABPP9916E **CANCEL_THREADS request is invalid because no selection criteria is specified.**

Explanation: No thread-selection criteria were specified for the CANCEL_THREADS request. One of the following must be specified: the ALL_THREADS parameter, the THREAD_TOKEN parameter, or one or more of the other thread-filtering parameters.

User response: Specify a thread-selection parameter in the ABPPARMS DD of the thread-cancelation job. Then run the job again.

ABPP9917E **Initialization parameter value is unknown:** ***parameter_name* = *parameter_value***

Explanation: An initialization parameter for the DB2 Utilities Enhancement Tool started task has an invalid value.

User response: See the DB2 Utilities Enhancement Tool documentation to determine the valid values for the specified initialization parameter. Then correct the value in your ABPOPTS file.

ABPP9918E **Initialization parameter value beyond range:** ***parameter_name* = *parameter_value***

Explanation: An initialization parameter for the DB2 Utilities Enhancement Tool started task has a value that is not within the allowable range for this parameter.

User response: See the DB2 Utilities Enhancement Tool documentation to determine the set of valid values for the specified initialization parameter. Then correct the parameter value in the ABPOPTS file.

ABPP9919E **Initialization parameter value is too long:** ***parameter_name* can be at most *parameter_max_length* characters**

Explanation: An initialization parameter for the DB2 Utilities Enhancement Tool started task is longer than the maximum length that is allowed for this parameter.

User response: Correct the parameter value in your initialization options member. Ensure that the value is not longer than the maximum length that is specified in this message text. Then run the job again.

ABPP9920E **Internal parser error: parser expected the address of the** ***control_block_name***

Explanation: An internal error occurred in the DB2 Utilities Enhancement Tool batch syntax parser or in the started task initialization options parser.

User response: Contact IBM Software Support.

ABPP9921W **Keyword *syntax_keyword* is unexpected. It will be ignored.**

Explanation: A keyword was found in an unexpected location in the command syntax. The keyword will be ignored.

User response: Correct the syntax and run the job again.

ABPP9922E **Initialization parameter contains nonnumeric characters:** ***parameter_name* = *parameter_value*.**

Explanation: An initialization parameter for the DB2 Utilities Enhancement Tool started task contains

nonnumeric characters. Only numeric characters are allowed.

User response: Correct the value and start the DB2 Utilities Enhancement Tool started task.

ABPP9925E Storage obtain failed. Module=<*module_name*>, storage area=<*storage_area_name*>, RC=<*return_code*>.

Explanation: The specified module failed while attempting to obtain the specified storage area.

User response: Increase the region size that is available to the DB2 Utilities Enhancement Tool program and run the product again. If the problem persists, contact IBM Software Support. Provide the support representative with the complete text of this message.

ABPP9927E An error was detected while attempting to open the input data set

Explanation: DB2 Utilities Enhancement Tool encountered an error while attempting to open the input data set for the DB2 utility job step.

User response: Check for other messages that are related to this error in the system log. Then correct the error and resubmit the job.

ABPP9928E I/O error when reading the input data set.

Explanation: DB2 Utilities Enhancement Tool encountered an I/O error when reading the input data set for the DB2 utility job step.

User response: Contact IBM Software Support.

ABPP9929E Buffer overflow error.

Explanation: While DB2 Utilities Enhancement Tool was parsing the DB2 utility job step, it detected a buffer overflow condition.

User response: Contact IBM Software Support.

ABPP9930E A syntax error was detected in the field specification for the field <*field_name*>.

Explanation: The field specification for the specified field in the DB2 LOAD utility job step contains a syntax error.

User response: Correct the field specification that is in error in the utility job step. Then run the utility again.

ABPP9931E Unbalanced parentheses detected in an INTO-TABLE specification.

Explanation: DB2 Utilities Enhancement Tool detected an unbalanced parenthesis (without a matching opening or closing parenthesis) in an INTO-TABLE specification of the LOAD utility job step.

User response: Correct this syntax error in the INTO-TABLE specification of the LOAD utility job step. Then run the utility again.

ABPP9932E The DELIMITED option is incompatible with the VALUEIF field selection criterion (START:END).

Explanation: While parsing the DB2 LOAD utility syntax, DB2 Utilities Enhancement Tool detected that the utility job step includes the DELIMITED option and a field selection criterion for the VALUEIF option that specifies a start:end byte position. This syntax is invalid. You cannot specify both the DELIMITED option and a VALUEIF field selection criterion that includes a start:end position in the same job step.

User response: Edit the LOAD utility job step to either remove the DELIMITED option or specify a field name instead of a start:end position in the field selection criterion for the VALUEIF option. Then run the utility again.

ABPP9933W DATABASE keyword is ignored if database name is specified with tablespace or indexspace.

Explanation: A database name is specified by the DATABASE parameter and also as part of the TABLESPACE or INDEXSPACE parameter value. The DATABASE parameter value is ignored.

User response: No action is required.

ABPP9934E An error was detected in the VALUEIF clause for field specification <*field_name*>.

Explanation: The field name in the field selection criterion of the VALUEIF clause does not match the field name of any field specification that is defined for the table to be loaded.

User response: In the INTO-TABLE portion of the LOAD utility job step, correct the field selection criterion of the VALUEIF clause or any field specification that is in error so that the field name in the field selection criterion of the VALUEIF clause matches the field name in a field specification. Then, run the utility job again.

ABPP9935E An operand of the DISCARDTO keyword is missing and must be specified.

Explanation: The DISCARDTO keyword was specified in the CHECK DATA utility syntax but one of the operands was not provided. Both operands are required for this keyword.

User response: Provide both operands for the DISCARDTO keyword and resubmit the job.

ABPP9936E An operand of the DISCARDSPACE keyword is missing and must be specified.

Explanation: The DISCARDSPACE keyword was specified in the CHECK DATA utility syntax but one of the operands was not provided. Both operands are required for this keyword.

User response: Provide both operands for the DISCARDSPACE keyword and resubmit the job.

ABPP9937E A zero value for an operand of the DISCARDSPACE keyword was specified.

Explanation: The DISCARDSPACE keyword was specified in the CHECK DATA utility syntax and one of the operands specified is a value of zero. A zero value for either the primary or secondary quantity is not allowed.

User response: Provide a valid value for both operands of the DISCARDSPACE keyword and resubmit the job. Valid values are -1 or 1 through 4,194,304.

ABPP9938E A value greater than the allowed maximum was specified in the DISCARDSPACE keyword.

Explanation: The DISCARDSPACE keyword was specified in the CHECK DATA utility syntax and one of the operands specified exceeded the maximum allowed.

User response: Provide a valid value for both operands of the DISCARDSPACE keyword and resubmit the job. Valid values are -1 or 1 through 4,194,304.

ABPP9939E Keyword PRESORT is incompatible with *&VARIABLE* **(where** *&VARIABLE* **can be one of the following values: FORMAT UNLOAD, FORMAT SQLDS, FORMAT INTERNAL, or NO FIELD SPECS).**

Explanation: PRESORT is not supported with the specified criteria.

User response: Correct the syntax and resubmit the job.

ABPP9940E Value exceeds maximum length for PRACTICE NAME <*practice_name***>.**

Explanation: In the DSNUTILB policy, the length of the specified practice name exceeds the maximum allowable length of 32 characters. This message provides the first 32 characters of the practice name that is in error.

User response: Correct the specified practice name in the DSNUTILB policy so that it does not exceed the maximum allowable length. After making the correction, resubmit the job. For more information about the PRACTICE element, see the product documentation.

ABPP9941E Attribute <*attribute_name***> is duplicated within a single element RULE.**

Explanation: In the DSNUTILB policy, the attribute displayed in the message text is duplicated within a single element RULE.

User response: In the DSNUTILB policy, delete the duplicated attribute in the single element RULE, and then resubmit the job.

ABPP9942E Invalid specification for keyword <*keyword***>.**

Explanation: The specified partition numbers are not valid. The partition numbers must be 1 - 4096. The first value must be lower than the second value.

User response: Correct the specified partition numbers. For information about specifying partition numbers, see the product documentation. After you make the correction in the POLICY, restart the started task.

ABPP9943E Keyword *keyword1* **is incompatible with keyword** *keyword2***.**

Explanation: Both of the specified keywords cannot be present in the load utility job input stream.

User response: Correct the syntax and resubmit the job.

ABPP9944E Value length of attribute <*attribute_name***> is more than <***attribute_length***> characters.**

Explanation: In the DSNUTILB policy, the length of the specified attribute value exceeds the maximum allowable length.

User response: Correct the attribute value. For more information about the attribute, see the product documentation.

ABPP9945W Invalid operand <*operand*>.

Explanation: The specified operand is only valid for load processing when you are running DB2 Version 9.1 and later.

User response: Remove the specified operand and then restart the job. For more information, see the section about load processing enhancements in the product user's guide.

ABPP9946E Only one table can be specified for load processing when you use the option <*keyword_name*>.

Explanation: The specified option is not supported for multiple tables in a LOAD statement.

User response: Specify only one table and then restart the job. For more information, see the product documentation.

ABPP9947I PRESORT was forced due to KEYWORD <*keyword_name*>.

Explanation: With the specified option, if PRESORT is not specified, LOAD processing continues as though it were.

User response: No action is required.

ABPP9948E Keyword <*keyword_name*> is incompatible with keyword <*keyword_name*>.

Explanation: The specified keywords cannot be used together in the syntax.

User response: Correct the syntax and resubmit the job.

ABPP9951E Keyword <*keyword_name*> is not supported when loading partition level SYSRECs.

Explanation: When PART *n* INDDN is specified in a LOAD utility statement, the specified keyword is are not supported.

User response: Remove the unsupported keyword and rerun the load utility job.

ABPS0000I DB2 Utilities Enhancement Tool <*product_version*>, FMID=<*product_fmid*>, COMPONENT ID=<*product_compid*>.

Explanation: This message provides the following information for your DB2 Utilities Enhancement Tool configuration: the version and release, FMID (an identifier for the release), and component ID. It is the first message issued to the SYSPRINT data set for the started task after the started task starts.

User response: No action is required.

ABPS0001I Started task initialization is in progress

Explanation: The initialization of the DB2 Utilities Enhancement Tool started task has begun.

User response: No action is required.

ABPS0002I Started task initialization is complete

Explanation: The initialization processing for the DB2 Utilities Enhancement Tool started task has successfully completed.

User response: No action is required.

ABPS0003I Started task termination is in progress

Explanation: Termination processing for the DB2 Utilities Enhancement Tool started task has begun.

User response: No action is required.

ABPS0004I Started task termination is complete

Explanation: The DB2 Utilities Enhancement Tool started task successfully completed termination processing.

User response: No action is required.

ABPS0007I TCB: <*tcb_address*> <*component_name*> - Component initialization is complete

Explanation: The initialization of the specified component completed successfully.

User response: No action is required.

ABPS0009I TCB: <*tcb_address*> <*component_name*> - Component termination is complete

Explanation: The termination of the specified component completed successfully.

User response: No action is required.

ABPS0010E TCB: <*tcb_address*> <*component_name*> - Component initialization failed.

Explanation: The initialization of the specified component was not successful.

User response: To determine the cause of the initialization failure, see the other messages that were issued for this component.

ABPS0012S TCB: <*tcb_address*> <*component_name*> received an unexpected post code. Post code=<*post_code*>.

Explanation: An internal error occurred.

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0013S **TCB: <***tcb_address***> <***component_name***> received an unexpected request code. Request code=<***request_code***>.**

Explanation: An internal error occurred.

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0014I **SVC installation is complete. SVC number = <***svc_number***>.**

Explanation: The installation of the DB2 Utilities Enhancement Tool supervisor call (SVC) was successful.

User response: No action is required.

ABPS0015I **Removing SVC. SVC number = <***svc_number***>.**

Explanation: The DB2 Utilities Enhancement Tool supervisor call (SVC) is in the process of being removed.

User response: No action is required.

ABPS0016E **SVC installation failed. SVC number = <***svc_number***>.**

Explanation: The installation of the DB2 Utilities Enhancement Tool supervisor call (SVC) was not successful.

User response: For more specific information about the SVC installation failure, see the messages that accompany this one.

ABPS0017S **SVC removal failed. SVC number = <***svc_number***>, RC = <***return_code***>, Reason = <***reason_text***>.**

Explanation: The removal of the DB2 Utilities Enhancement Tool supervisor call (SVC) was not successful. This message provides the return code and reason for this failure.

User response: Contact IBM Software Support. Provide Support with the complete text of this message, including the return code and reason text.

ABPS0018E **SVC installation failed. SVC number=<***svc_number***>, RC=<***return_code***>, reason=<***reason_text***>.**

Explanation: The installation of the DB2 Utilities Enhancement Tool supervisor call (SVC) was not successful. This message provides the return code and reason for the failure.

User response: Contact IBM Software Support. Provide Support with the complete text of this message, including the return code and reason text.

ABPS0019I **COMX:** *comx_address*, **COMI:** *comi_address*, **SVC EPA:** *svc_entry_point_address*, **MNTLEVEL:** *maintenance_level*

Explanation: This message is issued along with another message to provide diagnostic information to Support for resolving a problem.

User response: Provide this information to IBM Software Support when a Support representative requests it.

ABPS0020I **Logging has been started.**

Explanation: The DB2 Utilities Enhancement Tool started task has started writing log information to the ABPLOG table.

User response: No action is required.

ABPS0021I **Logging has been terminated.**

Explanation: The DB2 Utilities Enhancement Tool started task has stopped writing log information to the ABPLOG table.

User response: No action is required.

ABPS0022I **Auditing has been started.**

Explanation: The DB2 Utilities Enhancement Tool started task has started writing audit information to the ABPAUDIT table.

User response: No action is required.

ABPS0023I **Auditing has been terminated.**

Explanation: The DB2 Utilities Enhancement Tool started task has stopped writing audit information to the ABPAUDIT table.

User response: No action is required.

ABPS0024I **Tracing has been started.**

Explanation: The DB2 Utilities Enhancement Tool started task has started writing trace information to the internal trace table.

User response: No action is required.

ABPS0025I **Tracing has been terminated.**

Explanation: The DB2 Utilities Enhancement Tool started task has stopped writing trace information to the internal trace table.

User response: No action is required.

ABPS0080I Product initialization parameters:

Explanation: This message introduces a list of the initialization parameters that are defined for the DB2 Utilities Enhancement Tool started task. The list is printed when the started task starts.

User response: No action is required.

ABPS0081I *parm_name* **=** *parm_value_dec*

Explanation: This message provides the current decimal value for the specified started task initialization option. The message is issued only for options for which a decimal value is a valid value.

User response: No action is required.

ABPS0082I *parm_name* **=** *parm_value_char*

Explanation: This message provides the current integer value for the specified started task initialization parameter. This message is issued only for parameters for which an integer value is a valid value.

User response: No action is required.

ABPS0083S A value for the initialization parameter '*parm_name***' must be specified**

Explanation: The specified started task initialization parameter is not included in the initialization parameters file, or it has a value that is composed of only blanks. This parameter is required and must have a non-blank value.

User response: Ensure that this initialization parameter is in the initialization parameters file and is set to a non-blank value.

ABPS0085W '*parm_name***' must have a value from** *parm_min* **through** *parm_max***. Parameter defaulted to:** *parm_def***.**

Explanation: The value that is set for the specified DB2 Utilities Enhancement Tool started task initialization parameter is not within the allowable range of values for this parameter. As a result, the value will be changed to the default value for the parameter.

User response: Accept the default value, or specify a value that is within the allowable range of values for this parameter in the initialization options member.

ABPS0101I TCB: <*tcb_address***> Session created. SESS:** *session_token***-***session_number***-***session_type***-***session_job_name***-***session_job_ID***-***session_asid***-***session_user*

Explanation: The DB2 Utilities Enhancement Tool session was created. The session is identified by the information that is listed in this message after "SESS":

- *Session_token* is an internal session identifier.
- *Session_number* is a unique session identifier that is generated incrementally for each new session that is created.
- *Session_type* indicates whether the session is for a batch job (B), an ISPF user (I), the DSNUTILB intercept (U), or the ABPMAINT utility (M).
- *Session_job_name* is the name of the job that is associated with the session.
- *Session_job_ID* is the identifier for the job that is associated with the session
- *Session_asid* is the hexadecimal address space identifier for the user type (session type).
- *Session_user* is the user ID.

User response: No action is required.

ABPS0103I TCB: <*tcb_address***> Session terminated. SESS:***session_token***-***session_number***-***session_type***-***session_job_name***-***session_job_ID***-***session_asid***-***session_user*

Explanation: The DB2 Utilities Enhancement Tool session that was using the specified task control block (TCB) address space terminated. The attributes of this session are listed in this message after "SESS":

- *Session_token* is an internal session identifier.
- *Session_number* is a unique session identifier that is generated incrementally for each new session that is created.
- *Session_type* indicates whether the session is for a batch job (B), an ISPF user (I), the DSNUTILB intercept (U), or the ABPMAINT utility (M).
- *Session_job_name* is the name of the job that is associated with the session.
- *Session_job_ID* is the identifier for the job that is associated with the session.
- *Session_asid* is the hexadecimal address space identifier for the user type (session type).
- *Session_user* is the user ID.

User response: No action is required.

ABPS0200E TCB: <*tcb_address***> DB2 Call Attach Facility request <***caf_request***> failed, RC=<***return_code***>, RSN=<***reason_code***>.**

Explanation: The DB2 Call Attach Facility (CAF) returned the return code and reason code that is included in this message for the specified CAF request.

User response: Contact IBM Software Support. Provide Support with the return code and reason code that is included in this message.

ABPS0201S **TCB: <***tcb_address***> A Connect-to-DB2 request was received for** *db2_ssid***, but a connection already exists.**

Explanation: A request to connect to the specified DB2 subsystem was received. However, a connection to that subsystem is already established.

User response: Contact IBM Software Support.

ABPS0202E **TCB: <***tcb_address***>** *db2_error_msg*

Explanation: An error was encountered during an SQL or DB2 instrumentation facility interface (IFI) operation. This message contains the text of the message that the DB2 DSNTIAR message formatting routine issued when the error occurred.

A possible cause is that the started task does not have the proper authorization to perform the operation. The started task requires system administration authority (SYSADM) on all active subsystems in the data sharing group.

User response: For more information about the error, see the IBM DB2 messages documentation.

ABPS0203I **TCB: <***tcb_address***> Connection to DB2 was successful. SSID=***db2_ssid*

Explanation: DB2 Utilities Enhancement Tool successfully connected to the specified DB2 subsystem.

User response: No action is required.

ABPS0204I **TCB: <***tcb_address***> Disconnection from DB2 was successful. SSID=***db2_ssid*

Explanation: DB2 Utilities Enhancement Tool successfully disconnected from the specified DB2 subsystem.

User response: No action is required.

ABPS0205S **TCB: <***tcb_address***> STIMER SET failed. RC=<***return_code***>. Processing continues.**

Explanation: DB2 Utilities Enhancement Tool could not set a timing interval by using the STIMERM macro. Processing continues.

User response: Contact IBM Software Support.

ABPS0206S **TCB: <***tcb_address***> STIMER CANCEL failed. RC=<***return_code***>. Processing continues.**

Explanation: DB2 Utilities Enhancement Tool could not cancel a timing interval by using the STIMERM macro. Processing continues.

User response: Contact IBM Software Support.

ABPS0207E **TCB: <***tcb_address***> DB2 Instrumentation Facility request <***ifi_request***> failed, RC=<***return_code***>, RSN=<***reason_code***>, SSID=<***db2_ssid***>.**

Explanation: The specified request for the DB2 instrumentation facility interface (IFI) failed with the specified return code and reason code on the specified SSID.

User response: Contact IBM Software Support. Provide Support with the return code and reason code that is included in this message.

ABPS0208I **TCB: <***tcb_address***> Session: <***session_token***> - CANCEL THREAD issued for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool issued the CANCEL THREAD command for the thread that has the specified thread token value.

User response: No action is required.

ABPS0209E **TCB: <***tcb_address***> Connection to DB2 failed. SSID=***db2_ssid*

Explanation: DB2 Utilities Enhancement Tool could not connect to the DB2 subsystem that has the specified SSID.

User response: To determine the cause of the connection failure, see the message ABPS0202E in the message log. If you need assistance, contact IBM Software Support.

ABPS0210E **TCB: <***tcb_address***> Fatal error while processing the DB2 trace record:** *place_marker*

Explanation: A unrecoverable error occurred while DB2 Utilities Enhancement Tool was processing the DB2 trace record.

User response: Contact IBM Software Support.

ABPS0211I *db2_error_msg*

Explanation: The DB2 message formatting service DSNTIAR formatted the messages that follow this one in response to an action that was performed by an SQL or IFI operation.

User response: No action is required.

ABPS0212I **TCB: <***tcb_address***>. Lock data returned for ace token <***ace_token***>.**

Explanation: The DB2 instrumentation facility (IFI) returned lock data for the specified ace token.

User response: No action is required.

ABPS0213I **TCB: <***tcb_address***> Session: <***session_token***> - CANCEL THREAD NOBACKOUT was issued for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool issued the CANCEL THREAD command with the NOBACKOUT option for the thread that has the specified thread token value.

User response: No action is required.

ABPS0214E **Escalated Cancel is not supported for threads executing on a remote DB2 system.**

Explanation: The escalated cancel command is supported only for threads that are active on the DB2 system to which you connected. Use the normal DB2 cancel command to terminate threads that are active on other DB2 subsystems that are members of the same data-sharing group.

User response: No action is required.

ABPS0215I **TCB: <***tcb_address***> Session: <***session_token***> - ESCALATED THREAD CANCEL was issued for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool performed an escalated cancelation of the thread that has the specified thread token value. An escalated cancelation issues a command through the operator console to terminate the process that holds the thread.

User response: No action is required.

ABPS0216E **Escalated Cancel is not supported for connection type** *connection_type*

Explanation: The Escalated Cancel command is not supported for the specified connection type.

User response: No action is required.

ABPS0217I **ESCALATED THREAD CANCEL was issued for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool performed an escalated cancelation of the thread that has the specified thread token value. An escalated cancelation issues a command through the operator console to terminate the process that holds the thread.

User response: No action is required.

ABPS0218I **CANCEL THREAD was not issued because a unit of recovery exists for token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool did not issue a CANCEL THREAD command for the thread that has the specified thread token value because the NO BACKOUT option was specified as the cancel type. This option prevents the cancelation from occurring when an outstanding unit-of-recovery exists for a thread.

User response: No action is required.

ABPS0219I **CANCEL THREAD was not issued because unit of recovery status is unknown for token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool did not issue a CANCEL THREAD command for the thread that has the specified thread token value because the NO BACKOUT option was specified as the cancel type. This option prevents a cancelation from occurring when no unit-of-recovery information is available.

User response: No action is required.

ABPS0220I **TCB: <***tcb_address***> Session: <***session_token***> - CANCEL THREAD requested for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool received a CANCEL THREAD request for the thread that has the specified thread token value.

User response: No action is required.

ABPS0221E **TCB: <***tcb_address***> Session: <***session_token***> - CANCEL THREAD request failed security check for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool received a CANCEL THREAD request for the thread that has the specified thread token value. However, the request failed because it did not pass security-exit checking.

User response: No action is required.

ABPS0222E **TCB: <***tcb_address***> Session: <***session_token***> - pre-cancel exit denied cancel request.**

Explanation: DB2 Utilities Enhancement Tool received a CANCEL THREAD request for the thread that has the specified thread token value. However, the request failed because it did not pass pre-cancel exit checking.

User response: No action is required.

ABPS0223E **TCB: <***tcb_address***> Session: <***session_token***> - ESCALATED CANCEL not allowed by startup parm**

Explanation: An escalated cancelation cannot be performed because a started task initialization option is specified that does not allow this type of cancelation.

User response: No action is required.

ABPS0224E **TCB: <***tcb_address***> Session: <***session_token***> - CANCEL THREAD suppressed for ABP token** *thread_token*

Explanation: The DB2 CANCEL THREAD command and the escalated cancel command (z/OS operator Cancel command) is not supported for the current DB2 Utilities Enhancement Tool started task.

User response: No action is required.

ABPS0225E **CANCEL THREAD request failed security check for thread token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool received a CANCEL THREAD request for the thread that has the specified thread token value. However, the request failed because it did not pass security-exit checking.

User response: No action is required.

ABPS0226E **CANCEL THREAD request was denied by the pre-cancel exit for token** *thread_token*

Explanation: DB2 Utilities Enhancement Tool received a CANCEL THREAD request for the thread that has the specified thread token value. However, the request failed because it did not pass pre-cancel exit checking.

User response: No action is required.

ABPS0227E **TCB: <***tcb_address***> Session: <***session_token***> Start trace failed for instrumentation facility call**

Explanation: DB2 Utilities Enhancement Tool attempted to start the DB2 monitor trace facility prior to a call to the instrumentation facility interface. However, this attempt failed.

A possible cause is that the started task does not have the proper authorization to perform the operation. The started task requires system administration authority (SYSADM) on all active subsystems in the data sharing group.

User response: Contact IBM Software Support.

ABPS0228E **TCB: <***tcb_address***> Session: <***session_token***> Start trace failed for get_threads request**

Explanation: DB2 Utilities Enhancement Tool attempted to start the DB2 monitor trace facility for a get_threads request. However, this attempt failed.

User response: Contact IBM Software Support.

ABPS0229E **TCB: <***tcb_address***> Session: <***session_token***> Start trace failed for get_thread_detail request**

Explanation: DB2 Utilities Enhancement Tool attempted to start the DB2 monitor trace facility for a get_thread_detail request, but the attempt failed.

User response: Contact IBM Software Support.

ABPS0230E **DB2 CAF request <***db2_ssid***>, <***return_code***>, <***reason_code***>.**

Explanation: The DB2 Call Attach Facility (CAF) returned the return code and reason code that is included in this message for the specified CAF request.

User response: Contact IBM Software Support. Provide Support with the return code and reason code that is included in this message.

ABPS0231E **TCB: <***tcb_address***> Session: <***session_token***> Start trace failed for get_objects_referenced request**

Explanation: DB2 Utilities Enhancement Tool attempted to start the DB2 monitor trace facility for a get_objects_referenced request. However, this attempt failed.

User response: Contact IBM Software Support.

ABPS0232E **TCB: <***tcb_address***> Session: <***session_token***> -IP Address conversion error. RC=<***return_code***>, RSN=<***reason_code***>, *<***message_continuation_number***>*.**

Explanation: An internal error occurred during the conversion of a formatted IP address to a binary representation.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPS0233E **TCB: <***tcb_address***> Session: <***session_token***> -IP Address conversion error. RC=<***return_code***>, RSN=<***reason_code***>, *<***message_continuation_number***>*.**

Explanation: An internal error occurred during the conversion of a formatted IP address to a binary representation.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPS0234E ***<***message_continuation_number***>*** **<***ip_address***>.**

Explanation: DB2 Utilities Enhancement Tool failed to convert an IP address from an external text format to an internal binary format. The message ABPS0232E or ABPS0233E, which precedes this message, provides the return code and reason code for this error.

User response: Contact IBM Software Support.

ABPS0235W TCB: tcb_address. Authorization check for DB2 system *db2_ssid* **failed.**

Explanation: The started task authorization id has not been granted the minimum necessary authorization on the specified DB2 system.

User response: Grant the required authorization to the started task authorization id. See the user's guide for information about authorization requirements.

ABPS0236E TCB: tcb_address. Authorization for primary DB2 system *db2_ssid* **is insufficient.**

Explanation: The started task authorization id has not been granted the minimum necessary authorization on the primary DB2 system.

User response: Grant the required authorization to the started task authorization id. See the user's guide for information about authorization requirements.

ABPS0300E TCB: <*tcb_address***> IEAVRLS Pause Release failed, RC=<***return_code***>.**

Explanation: The IEAVRLS Pause Release Service failed with the specified return code.

User response: Contact IBM Software Support. Provide Support with the return code from this message.

ABPS0301E TCB:<*tcb_address***>. Session=<***session_token***>. Unable to return result.**

Explanation: The specified DB2 Utilities Enhancement Tool session could not return the results of an operation to the user.

User response: For more information about this error, see the other messages that were issued for the specified task control block (TCB) and session. If you need assistance, contact IBM Software Support.

ABPS0302E TCB: <*tcb_address***> ABPSRSLT bad parms, Session: <***session_token***>, FBUF=***fbuf_address***, UBUF=***ubuf_address*

Explanation: An internal error occurred. DB2 Utilities Enhancement Tool invoked the ABPSRSLT results processor by using an invalid FBUF or UBUF address pointer.

User response: Contact IBM Software Support.

ABPS0303I TCB: <*tcb_address***> Failure to obtain ALET, Session:***session_token*

Explanation: An internal error occurred. DB2 Utilities Enhancement Tool could not obtain the ALET token to facilitate cross-memory addressing.

User response: Contact IBM Software Support.

ABPS0304I TCB: <*tcb_address***> STOKEN release failure**

Explanation: An internal error occurred. DB2 Utilities Enhancement Tool could not release the STOKEN token, which is involved in cross-memory addressing.

User response: Contact IBM Software Support.

ABPS0305S TCB: <*tcb_address***> Session failed. SESS:***session_token csect_name/offset_value/ variable_value/variable_name*

Explanation: An internal error occurred. DB2 Utilities Enhancement Tool failed to validate a cross-memory address. This failure probably occurred because a client address space terminated abnormally.

User response: Contact IBM Software Support.

ABPS0306E TCB: <*tcb_address***> SQL Error occurred. Module:** *module_name offset_value***.**

Explanation: An SQL error occurred.

User response: Review the information in the ABPS0202E messages that follow this one for detailed information about the error. Also see the DB2 messages documentation to determine the reason for the error. If you need assistance, contact IBM Software Support. Provide Support with the TCB address and module name that is included in this message.

ABPS0307E TCB: <*tcb_address***> SRB Processing returned, RC=<***return_code***>, RSN=<***reason_code***>, RSN1=<***extended_reason_code***>.**

Explanation: SRB processing returned the specified error codes.

User response: Contact IBM Software Support. Provide the support representative with the return code from this message.

ABPS0308W TCB: <*tcb_address***> Unable to determine the index space name for DBID:** *database_id* **OBID:** *object_id*

Explanation: DB2 Utilities Enhancement Tool could not determine the index space name for the DBID and OBID that are identified in this message.

User response: Contact IBM Software Support.

ABPS0309W TCB: <*tcb_address***> Unable to determine the table space name for DBID:** *database_id* **OBID:** *object_id*

Explanation: DB2 Utilities Enhancement Tool could not determine the table space name for the DBID and OBID that are identified in this message.

User response: Contact IBM Software Support.

ABPS0310W TCB: <*tcb_address***> Unable to access ABPLOG table**

Explanation: The ABPLOG table was not found. Therefore, DB2 Utilities Enhancement Tool cannot write messages to this table. The table should have been created on the primary subsystem during customization.

User response: Review the DB2 Utilities Enhancement Tool customization procedures. Make sure that you created the ABPLOG table by using the DDL member that Tools Customizer created for your primary subsystem. Also make sure that the DB2_SSID option in the your *product_id*OPTS member specifies the DB2 subsystem where the ABPLOG table is located.

ABPS0311W TCB: <*tcb_address***> Unable to access ABPAUDIT table**

Explanation: The ABPAUDIT table was not found. Therefore, the product cannot write audit information to this table. The table should have been created on the primary subsystem during customization.

User response: Review the product customization procedures. Make sure that you created the ABPAUDIT table by using the member that Tools Customizer created for your primary subsystem. Also make sure that the DB2_SSID option in the ABPOPTS member specifies the DB2 subsystem where the ABPAUDIT table is located.

ABPS0400S Task Manager initialization failed

Explanation: The DB2 Utilities Enhancement Tool task management component failed during started task initialization. Processing will terminate.

User response: For more information about this error, see the other messages that were issued just prior to this message. If you need assistance with resolving this problem, contact IBM Software Support.

ABPS0401S Component ID=*component_id* **Component not found in the MEPL table**

Explanation: An internal error occurred.

User response: Contact IBM Software Support.

ABPS0402S Attach failed. Program=<*program_name***>, RC=<***return_code***>, RSN=<***reason_code***>.**

Explanation: An internal error occurred.

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0403W TCB: <*tcb_address***>, Detach failed. RC=<***return_code***>, RSN=<***reason_code***>.**

Explanation: An internal error occurred.

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0404S TCB: <*tcb_address***> Subtask failed. Termination ECB:** *event_control_block***.**

Explanation: An internal error occurred.

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0405S TCB: <*tcb_address***> Subtask unexpectedly posted init ECB. Initialization ECB:** *event_control_block***.**

Explanation: An internal error occurred that is related to the specified event control block (ECB).

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0406S TCB: <*tcb_address***> Subtask failed during initialization. Termination ECB:** *event_control_block***.**

Explanation: An internal error occurred that is related to the specified event control block (ECB).

User response: Contact IBM Software Support. Provide Support with the complete text of this message.

ABPS0407W Task manager received an unexpected command code. Command code=<*command_code***>.**

Explanation: An internal error occurred.

User response: Contact IBM Software Support.

Provide Support with the complete text of this message.

ABPS0408S No DB2 task started. Processing will be terminated.

Explanation: An internal error occurred.

User response: Contact IBM Software Support.

ABPS0409W Cannot connect to the primary DB2 subsystem <*db2_subsystem*> as required.

Explanation: DB2 Utilities Enhancement Tool could not connect to the DB2 subsystem that is necessary for logging and auditing functions because that subsystem is not active. Processing continues; however, logging and auditing cannot be performed.

User response: Ensure that the DB2 subsystem that is specified in the initialization options member is started and available for use by DB2 Utilities Enhancement Tool.

ABPS0410E Primary DB2 subsystem <*db2_subsystem*> does not exist. Processing will be terminated.

Explanation: DB2 Utilities Enhancement Tool could not connect to the DB2 subsystem that is necessary for logging and auditing functions because that subsystem does not exist.

User response: Ensure that the DB2 subsystem that is specified in the initialization options member exists and is available for use by DB2 Utilities Enhancement Tool.

ABPS0500I TCB: <*tcb_address*> Session: <*session_token*> SSID: *db2_ssid* BLOCKER ID: *thread_blocker_id* **message_continuation_number**

Explanation: A thread-blocking operation that has the specified blocker ID has been initiated. The messages that follow this one identify the operations and DB2 object status changes.

User response: No action is required.

ABPS0501I **message_continuation_number** **(***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number* **OLD:** *old_status* **NEW:** *new_status*

Explanation: DB2 Utilities Enhancement Tool changed the status of the specified DB2 object while performing a thread-blocking operation. This message provides the old status and the new status for the object. The message ABPS0500I provides the blocker ID for the thread-blocking operation.

User response: No action is required.

ABPS0502I **message_continuation_number** **(***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number*

Explanation: DB2 Utilities Enhancement Tool failed to perform a thread-blocking operation on the specified DB2 object because threads were already being blocked on that object. The message ABPS0503I provides the blocker ID for the thread-blocking operation.

User response: No action is required.

ABPS0503W **message_continuation_number** **Object already blocked by blocker ID** *blocker_id*

Explanation: DB2 Utilities Enhancement Tool failed to perform the thread- blocking operation that has the blocker ID specified in the message ABPS0500I and that was attempting to block threads on the DB2 object identified in the message ABPS0502I. Threads on that object were already being blocked by a previous thread-blocking operation that has the blocker ID specified in this message. The processing of the current thread-blocking operation continues because the ON_FAILURE (CONTINUE) parameter is specified for the job step.

User response: No action is required.

ABPS0504E **message_continuation_number** **The blocker ID specified was not found for delete.**

Explanation: An attempt was made to delete information for the specified thread-blocker ID from the DB2 Utilities Enhancement Tool object status table (ABPOBJSTAT). This attempt failed because the table contained no information for that blocker ID.

User response: Make sure that the blocker ID that is specified in the PARM in the EXEC statement of the thread-blocker job step is spelled correctly.

ABPS0505E **message_continuation_number** **Thread blocker ID is already in use.**

Explanation: DB2 Utilities Enhancement Tool failed to perform a thread-blocking operation on a DB2 object because the blocker ID is already in use. The blocker ID is specified in the message ABPS0500I.

User response: Specify a unique blocker ID.

ABPS0506W **message_continuation_number** **(***cancel_specification_number***) No objects could be resolved for cancel specification.**

Explanation: An attempt to resolve the database and space objects for a thread-blocking action under this cancel specification failed. The objects were not found in the DB2 catalog.

User response: Make sure that the object names that are specified in the cancel specification are spelled correctly. If any wildcard patterns are specified, make sure that they will resolve to the correct DB2 objects.

ABPS0507I ******message_continuation_number***** Blocker ID deleted.**

Explanation: The specified blocker ID was deleted from the object status table.

User response: No action is required.

ABPS0508I ******message_continuation_number***** No objects to reset for this blocker ID.**

Explanation: DB2 Utilities Enhancement Tool failed to find any rows in the object status table (ABOBJSTAT) that matched the blocker ID.

User response: No action is required.

ABPS0509I ******message_continuation_number***** Reset status processing initiated.**

Explanation: DB2 Utilities Enhancement Tool initiated processing to reset the object status in response to a previous error condition.

User response: No action is required.

ABPS0510I ******message_continuation_number***** Thread blocker operation is** *thread_blocker_operation*

Explanation: This message identifies the current thread-blocking operation.

User response: No action is required.

ABPS0511E ******message_continuation_number***** (***cancel_specification_number***) No objects could be resolved for cancel specification.**

Explanation: An attempt to resolve the database and space objects for a thread-blocking action under this cancel specification failed. The objects were not found in the DB2 catalog. Processing is terminated because the ON_FAILURE (TERMINATE) parameter was specified for the job step.

User response: Make sure that the object names that are specified in the cancel specification are spelled correctly. If any wildcard patterns are specified, make sure that they will resolve to the correct DB2 objects.

ABPS0512E ******message_continuation_number***** Object already blocked by blocker ID** *blocker_id*

Explanation: DB2 Utilities Enhancement Tool failed to perform the thread-blocking operation that has the blocker ID specified in the ABPS0500I message and that was attempting to block threads on the DB2 object identified in the ABPS0502I message. Threads on that object were already being blocked by a previous thread-blocking operation. This message presents the blocker ID of the previous thread-blocking operation. The processing of the current thread-blocking operation was terminated because the ON_FAILURE (TERMINATE) parameter is specified for the job step.

User response: Determine if the current thread-blocking operation is in conflict with the previous thread-blocking operation. If a conflict exists, wait until an ALLOW_THREADS or DELETE_BLOCKERID job step ends the previous thread-blocking operation. If a conflict does not exist, change the ON_FAILURE parameter value to CONTINUE for the current thread-blocking operation and then resubmit the job.

ABPS0513I ******message_continuation_number***** (***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number* **Object not found**

Explanation: DB2 Utilities Enhancement Tool could not change the status of the specified DB2 object because the object no longer exists. The message ABPS0500I provides the blocker ID for the thread-blocking operation.

User response: No action is required.

ABPS0514I ******message_continuation_number***** (***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number* **OLD:** *old_status*

Explanation: DB2 Utilities Enhancement Tool did not change the status of the specified DB2 object while performing a thread-blocking operation because the object was already in the desired state. The message ABPS0500I provides the blocker ID for the thread-blocking operation.

User response: No action is required.

ABPS0515I ******message_continuation_number***** Thread blocker is suppressed for DB2 system object database_name.**

Explanation: The thread blocker operation is suppressed for the following DB2 system databases: DSNDB01, DSNDB06, and DSNDB07.

User response: No action is required.

ABPS0516W ******message_continuation_number***** (***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number* **Partition number is invalid.**

Explanation: The thread blocker operation detected an

invalid partition number specification.

User response: No action is required.

ABPS0517E ******message_continuation_number***** (***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number* **Partition number is invalid.**

Explanation: The thread blocker operation detected an invalid partition number specification on a DB2 Version 7 system. The thread blocker operation cannot continue.

User response: Correct the partition specification and rerun the job.

ABPS0518E ******message_continuation_number***** (***cancel_specification_number***) DB:** *database_name* **SP:** *space_name* **PART:** *partition_number* **Partition number is invalid.**

Explanation: The thread blocker operation detected a partition number specified for a non-partitioned space. The thread blocker operation cannot continue.

User response: Correct the partition specification and rerun the job.

ABPS0519I **Thread blocker is suppressed for DB2 Utilities Enhancement Tool configuration object** *DB: database_name SP: space_name***.**

Explanation: The thread blocker operation is suppressed for the DB2 Utilities Enhancement Tool configuration database.

User response: No action is required.

ABPS0520E ******message_continuation_number***** Userid** *user_id* **denied access to blocker operation by security exit.**

Explanation: The security exit for the DB2 Utilities Enhancement Tool configuration prevented the specified user from performing a thread-blocker operation.

User response: To perform a thread-blocker operation, the user must be provided with the proper authority under the security exit.

ABPS0521I **Thread blocker is suppressed for the DB2 object with type TEMP or WORKFILE:** *database_name*

Explanation: Because a database that is defined as WORKFILE or TEMP cannot be started in RO or UT status, thread-blocker operations must not be attempted for such an object. Therefore, the thread-blocker operation is suppressed for objects that are in a database with a value of "W" or "T" in the SYSIBM.SYSDATABASE TYPE column.

User response: No action is required.

ABPS0600I **DSNUTILB interception for DB2 SSID=***DB2_ssid* **is enabled.**

Explanation: DSNUTILB interception services have been enabled for the specified DB2 subsystem.

User response: No action is required.

ABPS0601I **DSNUTILB interception for DB2 SSID=***DB2_ssid* **is disabled.**

Explanation: DSNUTILB interception services have been disabled for the specified DB2 subsystem.

User response: No action is required.

ABPS0602W **DSNUTILB interception for DB2 SSID=***DB2_ssid* **not enabled. Interception being performed by ABPID=***product_id***.**

Explanation: DSNUTILB interception services were not enabled for the DB2 subsystem that is specified in this message because another DB2 Utilities Enhancement Tool system was already providing interception services for it.

User response: Verify that the list of DB2 subsystems in the DSNUTILB interception policy is correct. Only one DB2 Utilities Enhancement Tool system can provide interception services for a specific DB2 subsystem at a time.

ABPS0603W **DSNUTILB interception for DB2 SSID=***DB2_ssid* **not enabled, product cannot connect to the subsystem**

Explanation: DSNUTILB interception services were not enabled for the DB2 subsystem that is indicated in this message because DB2 Utilities Enhancement Tool cannot connect to that DB2 subsystem.

User response: Verify that the list of DB2 subsystems that is specified in the DSNUTILB intercept policy is correct. Only one DB2 Utilities Enhancement Tool system can provide interception services for a specific DB2 subsystem at one time. DB2 Utilities Enhancement Tool must have a properly bound plan on the DB2 subsystem for which it will provide interception services.

ABPS0604W **DSNUTILB interception for DB2 SSID=***DB2_ssid* **not enabled, DB2 subsystem is not active.**

Explanation: DSNUTILB interception services were not enabled for the DB2 subsystem that is indicated in this message because the subsystem is inactive.

User response: Verify that the list of DB2 subsystems specified in the DSNUTILB interception policy is correct. Only one DB2 Utilities Enhancement Tool system can provide interception services for a specific DB2 subsystem at a time.

ABPS0605W DSNUTILB interception for *db2_ssid* not enabled, insufficient authority.

Explanation: DSNUTILB interception services were not enabled for the DB2 subsystem that is indicated in this message because DB2 Utilities Enhancement Tool has insufficient authority on that DB2 subsystem.

User response: Grant the required authorization to the started task authorization id. See the user's guide for information about authorization requirements.

ABPS0606I DB2 SSID=*db2_ssid* has DB2 Sort enabled.

Explanation: DB2 Sort is either enabled (YES) or not enabled (NO) for the specified DB2 subsystem.

User response: No action is required.

ABPS0607I TCB: <*tcb_address*>, DB2 subsystem <*db2_ssid*>, startup detected.

Explanation: The DB2 Utilities Enhancement Tool started task detected that a DB2 system that is referenced in the policy has started.

User response: No action is required.

ABPS0608W TCB: <*tcb_address*>: Count of DB2 systems exceeds 256. Startup detection disabled for SSID <*db2_ssid*>.

Explanation: The DB2 Utilities Enhancement Tool started task detected that the number of DB2 subsystems referenced by the policy exceeds the maximum of 256.

User response: Refine the policy to reduce the number of referenced DB2 subsystems.

ABPS0609I TCB: <*tcb_address*>: DB2 system <*db2_ssid*> is the primary subsystem for this instance.

Explanation: The DB2 Utilities Enhancement Tool started task is using the DB2 system as its primary subsystem. All log and audit records are inserted using the connection established for this DB2. If this DB2 system is stopped while the DB2 Utilities Enhancement Tool started task is running, the started task will terminate.

User response: No action is required.

ABPS0700I TCB *tcb_address* SESSION REPORT *message_continuation_number*

Explanation: A session report has been initiated. The messages that follow represent details about currently active sessions.

User response: No action is required.

ABPS0701I *message_continuation_number* SESS: *Session_token* Session_number *Session_type Session_job_name Session_job_ID Session_asid Session_user*

Explanation: Details of a product session. The session is identified by the information that is displayed in this message after SESS:

- *Session_token* is an internal session identifier.
- *Session_number* is a unique session identifier that is generated incrementally for each new session that is created.
- *Session_type* indicates whether the session is for a batch job (B), an ISPF user (I), the DSNUTILB intercept (U), or the ABPMAINT utility (M).
- *Session_job_name* is the name of the job that is associated with the session.
- Session_job_ID is the identifier for the job that is associated with the session.
- *Session_asid* is the hexadecimal address space identifier for the user type (session type).
- *Session_user* is the user ID.

User response: No action is required.

ABPS0702I *message_continuation_number* STATUS: *session_status*

Explanation: Session status.

User response: No action is required.

ABPS0703I *message_continuation_number* STARTED: *session_start_time*

Explanation: Date and time when session was started.

User response: No action is required.

ABPS0704I *message_continuation_number* No active sessions found

Explanation: No active sessions were found.

User response: No action is required.

ABPS0804W The trace table is too small. Tracing will be disabled. Required minimum size= *trace_table_minimum_size*, Requested size= *trace_table_requested_size*

Explanation: The size of the trace table is too small to

perform internal tracing. Tracing will be disabled, but product operations will continue.

User response: Increase the size of the trace table to at least the minimum size that is indicated in this message.

ABPS0805W The trace table entry is larger than the trace table. Trace table size= *trace_table_size*, **Trace entry size=** *trace_table_entry_size*

Explanation: The size of the trace information entry is larger than the size of the trace table. The entry cannot be recorded in the trace table.

User response: Increase the size of the trace table. If the problem persists, contact IBM Software Support.

ABPS0806I The *user_exit_type* **User Exit** *user_exit_name* **is now in use.**

Explanation: The specified user exit is in use.

User response: No action is required.

ABPS0807S A severe error occurred while attempting to load the *exit_type* **user exit** *exit_name*

Explanation: DB2 Utilities Enhancement Tool started task encountered a severe error when attempting to load the specified user exit.

User response: Ensure that the following requirements are met: 1) the exit is properly assembled and linked, 2) the exit resides in a STEPLIB-concatenated load library that is accessible to the DB2 Utilities Enhancement Tool started task, and 3) the exit name is correctly specified in the started task initialization options member.

ABPS0808S A severe error occurred within *exit_type* **user exit** *exit_name*, **FUNC=***exit_function*

Explanation: The DB2 Utilities Enhancement Tool started task encountered a severe error within the specified user exit.

User response: An MVS™ SVC dump has been produced to help you diagnose the problem with the user exit. After you correct the problem, assemble and link the exit. Then restart DB2 Utilities Enhancement Tool.

ABPS0809S A severe internal error occurred preparing to drive the *exit_type* **user exit** *exit_name*, **FUNC=***exit_function*

Explanation: The DB2 Utilities Enhancement Tool started task encountered a severe internal error while preparing to run the specified user exit.

User response: Contact IBM Software Support.

ABPS0810I The *user_exit_type* **User Exit** *user_exit_name* **is now inactive.**

Explanation: The specified user exit is no longer active.

User response: No action is required.

ABPS0811S The <*user_exit_type***> user exit <***user_exit_name***> FUNC=<***user_exit_func***> RC=12. The started task is terminating.**

Explanation: The DB2 Utilities Enhancement Tool started task received the return code RC=12 from the specified user exit. As a result, the started task is terminating.

User response: Identify and correct the problem that caused the user exit to issue RC=12. Then restart the DB2 Utilities Enhancement Tool started task.

ABPS0812I MODULE LEVEL DATE TIME EPA RREPA CC F1 F2 F3 SEQ

Explanation: This message displays the fields in the Module Entry Point List (MEPL) control block.

User response: No action is required.

ABPS0813I <*module_name***>, <***maintenance_level***>, <***assembly_date***>, <***assembly_time***>, <***entry_point_address***>, <***rr_entry_point_address***>, <***component_code***>, <***flag_byte_1***>, <***flag_byte_2***>, <***flag_byte_3***>, <***sequence_number***>.**

Explanation: This message displays the data in the fields of the Module Entry Point List (MEPL) control block.

User response: No action is required.

ABPS0814I Command issued: *command_text*

Explanation: This message identifies the DB2 Utilities Enhancement Tool operator command that was issued from the z/OS console.

User response: No action is required.

ABPS0815E Unrecognized command

Explanation: An unknown operator command was issued to the DB2 Utilities Enhancement Tool started task.

User response: Specify a valid DB2 Utilities Enhancement Tool command.

ABPS0816E Invalid keyword provided for command: *command_name*

Explanation: An invalid keyword was provided for the DB2 Utilities Enhancement Tool command that is specified in this message.

User response: Specify a valid keyword for the command. For the correct syntax, see the DB2 Utilities Enhancement Tool documentation.

ABPS0817I *command_scope* **DSNUTILB intercept status is:** *dsnutilb_intercept_status*

Explanation: This message indicates either the local DSNUTILB intercept status for the started task or the global DSNUTILB intercept status for the entire z/OS image.

User response: No action is required.

ABPS0818I *help_text*

Explanation: This message presents the output from the HELP console command that was issued for the DB2 Utilities Enhancement Tool started task. This command lists all console commands that are supported for the started task.

User response: No action is required.

ABPS0819E Trace table size is zero. Trace table display is not possible.

Explanation: A SNAP of the DB2 Utilities Enhancement Tool trace table was requested, but no trace table exists. The trace table does not exist because the trace table size option is set to zero. Therefore, the trace data cannot be displayed.

User response: If you want to be able to record DB2 Utilities Enhancement Tool internal trace data, set the trace table size to a non-zero value in the started task initialization options member.

ABPS0820W A display of the trace table is already in progress.

Explanation: A SNAP of the DB2 Utilities Enhancement Tool trace table is already in progress. Consequently, this additional request is ignored.

User response: If you want to display the DB2 Utilities Enhancement Tool internal trace table again, wait for the current display request to complete.

ABPS0821I Trace table display is complete.

Explanation: The requested display of the DB2 Utilities Enhancement Tool internal trace table has completed.

User response: No action is required.

ABPS0822I DB2SSID=*db2_ssid* **DB2VER=***db2_version* **ABPID=***configuration_id* **DSNUTILB interception is** *DSNUTILB_interception_status*

Explanation: This message presents the DSNUTILB intercept status for the specified DB2 subsystem.

User response: No action is required.

ABPS0823E Address contains invalid hex digits

Explanation: An invalid address was specified in the console command. The address contained invalid characters. An address must be an 8-digit hexadecimal number that is composed of only the characters 0 through 9 and A through F.

User response: Specify a valid hexadecimal address for the command.

ABPS0824E Address is not for an active session

Explanation: The address that was specified in the TERMINATE SESSION console command does not reference an active session. The session might have already terminated, or the address might have been entered incorrectly.

User response: Verify that the session address was entered correctly. If the session address was incorrect, reissue the TERMINATE SESSION command with a valid session address. If the address was correct, the session already terminated and you do not need to take further action.

ABPS0830I DSNUTILB Intercept Policy:

Explanation: This message introduces the DSNUTILB intercept policy. The policy details are presented in the messages that follow this one.

User response: No action is required.

ABPS0831I DB2 SSID: *db2_ssid* **ACTION:** *action* | **ACTION:** *VRUPDATE - SUBMIT_FROM_SERVER*

Explanation: This message identifies the section of the DSNUTILB intercept policy that is for the specified DB2 subsystem and defined ACTION to perform.

If SUBMIT_FROM_SERVER="NO" or is omitted from the policy, the message ABPS0831I states "ABPS0831I DB2 SSID: *db2_ssid* ACTION: *action*."

If SUBMIT_FROM_SERVER="YES" is specified in the policy, the message ABPS0831I states "ABPS0831I DB2 SSID: *db2_ssid* ACTION: *VRUPDATE - SUBMIT_FROM_SERVER*."

User response: No action is required.

ABPS0832I **Rule type:** *rule_type*

Explanation: This message identifies an INCLUDE or EXCLUDE rule in the DSNUTILB intercept policy.

User response: No action is required.

ABPS0833I *rule_number delimiter rule_element_type delimiter rule_element_data*

Explanation: This message presents a RULE element that is specified in the DSNUTILB intercept policy.

User response: No action is required.

ABPS0834I **DSNUTILB intercept is inactive.**

Explanation: The command was not processed because the DSNUTILB intercept was turned off in the initialization options.

User response: No action is required.

ABPS0835I **Active PRACTICE:** *practice_name*

Explanation: This message indicates the name of the active PRACTICE of the DSNUTILB intercept policy.

User response: No action is required.

ABPS0836I **DB2 Utilities Enhancement Tool started task practice report**

Explanation: This message presents the output from the LIST PRACTICE or DISPLAY PRACTICE console command that was issued for the DB2 Utilities Enhancement Tool started task.

User response: No action is required.

ABPS0840E **TCB: <***tcb_address***>. Error on INSERT to table SYSAUTO.UTILITYRUNS_HISTORY.**

Explanation: ACTION=AUTO_DIRECTOR was specified in the DB2 Utilities Enhancement Tool policy, but the product encountered an error while attempting to insert a row into the utility execution history table.

User response: See additional formatted SQL error messages in the DB2 Utilities Enhancement Tool SYSPRINT.

ABPS0841W **TCB: <***tcb_address***>. DB2 Autonomics Director collection disabled. BBY$NMIC bad offset to data.**

Explanation: The module BBY$NMIC that was found in the DB2 Utilities Enhancement Tool started task contains an offset to the data structure that does not point to a valid version. DB2 Autonomics Director utility history collection is disabled.

User response: Contact IBM Software Support.

ABPS0898D **DEBUG: Field:** *field_name* **Value:** *field_value*

Explanation: DB2 Utilities Enhancement Tool could not connect to the DB2 subsystem that is specified in the initialization options member because that subsystem is not active.

User response: Ensure that the DB2 subsystem that is specified in the initialization options member is started and available for use by DB2 Utilities Enhancement Tool.

ABPS0899D **DEBUG: P Len:** *plan_length* **P Name:** *plan_name* **Q Len:** *qual_length* **N Len:** *name_length* **IN1:** *type_1* **IN2:** *type_2* **Flag:** *flag*

Explanation: DB2 Utilities Enhancement Tool could not connect to the DB2 subsystem that is specified in the initialization options member because that subsystem is not active.

User response: Ensure that the DB2 subsystem that is specified in the initialization options member is started and available for use by DB2 Utilities Enhancement Tool.

ABPS0900E **The product is not APF-authorized and is terminating.**

Explanation: The load library for the product started task is not APF-authorized, as required. Consequently, the product is terminating.

User response: APF-authorize the load library for the started task, and then start the product again.

ABPS0901S **RVT locate or allocate operation failed.**

Explanation: The product could not locate or allocate its RVT control block.

User response: Contact IBM Software Support.

ABPS0902S **DB2 Utilities Enhancement Tool started task ESTAE entered, S<***system_completion_code***>, U<***user_completion_code***>.**

Explanation: The main task of the DB2 Utilities Enhancement Tool started task encountered an error. A dump has been generated.

User response: Review the dump data to diagnose and resolve the problem. If you need assistance, contact IBM Software Support.

ABPS0903S ESTAE processing completed

Explanation: DB2 Utilities Enhancement Tool finished generating a dump for the error that was encountered by the main task of the started task.

User response: Review the dump data to diagnose the problem. If you need assistance, contact IBM Software Support.

ABPS0904S Started task subtask ESTAE entered, S<*system_completion_code*>, U<*user_completion_code*>.

Explanation: A subtask of the DB2 Utilities Enhancement Tool started task encountered an error. A dump has been generated.

User response: Review the dump data to diagnose and resolve the problem. If you need assistance, contact IBM Software Support.

ABPS0905S User exit for the started task encountered an error. A dump was created. System RC=<*system_completion_code*>, user RC=<*user_completion_code*>.

Explanation: A security exit, pre-cancel exit, or post-cancel exit that you specified for the DB2 Utilities Enhancement Tool started task encountered an error when it ran. A dump has been generated for diagnostic use.

User response: Review the dump data to resolve the problem with the user exit. The names of all user exits are specified in the started task initialization options member. If you need assistance, contact IBM Software Support.

ABPS0906S SVC removal failed

Explanation: DB2 Utilities Enhancement Tool could not remove its supervisor call (SVC) when the started task stopped.

User response: Contact IBM Software Support.

ABPS0907S ABPGMODL Load Failed for MEPL=*mepl_name*.

Explanation: An internal error occurred during the initialization of the product started task.

User response: Make sure that the JCL for the started task points to the proper STEPLIB. If the problem persists, contact IBM Software Support.

ABPS0908S ABPGMODL Load Failed for MEPL entry=*mepe_name*.

Explanation: An internal error occurred during the initialization of the product started task.

User response: Make sure that the JCL for the started task points to the proper STEPLIB. If the problem persists, contact IBM Software Support.

ABPS0909S Started task subtask ESTAE entered, system RC=<*system_completion_code*>, user RC=<*user_completion_code*>.

Explanation: A subtask of the DB2 Utilities Enhancement Tool started task encountered an error. A dump will be created to help you diagnose the problem.

User response: Review the dump data to diagnose the problem. If you need assistance, contact IBM Software Support.

ABPS0910E A job name conflict with a started task has been identified. The product is terminating.

Explanation: The job name for the DB2 Utilities Enhancement Tool started task conflicts with the job name for another started task on the z/OS system. Consequently, the product is terminating.

User response: Either change the name of the DB2 Utilities Enhancement Tool started task or the name of the started task that is in conflict, and then start DB2 Utilities Enhancement Tool again.

ABPS0911E A job name conflict with a batch job has been identified. The product is terminating.

Explanation: The job name for the DB2 Utilities Enhancement Tool started task conflicts with the name of a batch job on this z/OS system. Consequently, the product is terminating.

User response: Either change the name of the DB2 Utilities Enhancement Tool started task or the name of the batch job that is in conflict, and then start DB2 Utilities Enhancement Tool again.

ABPS0912E ABPID already in use. Terminating.

Explanation: Another DB2 Utilities Enhancement Tool started task that is running on the z/OS system has the same identifier. Each product started task must have a unique identifier. Therefore, the started task for which this message was issued is terminating.

User response: Make sure that every product started task that runs concurrently on your system has a unique identifier.

ABPS0913I **ESTAE SDUMPX call RC=<*short_system_return_code*>, RS=<*short_system_reason_code*>.**

Explanation: During ESTAE processing, a call to the z/OS SDUMPX facility returned the displayed return code and reason code.

User response: If RC=08, review the reason code in the appropriate SDUMPX documentation. Then make any changes to Dump Services that are needed to obtain proper diagnostic dumps. If you need assistance, contact IBM Software Support.

ABPS5100I **TCB: <***tcb_address***> Session: <***session_token***> SSID:** *db2_ssid* **DSNUTILB utility id :** *utility_id* ******message_continuation******

Explanation: A DSNUTILB intercept operation was initiated for the specified DSNUTILB utility ID. The messages that follow this one identify the intercept operation and present data associated with it.

User response: No action is required.

ABPS5101I ******message_continuation_number****** **DSNUTILB intercept operation is** *DSNUTILB_intercept_operation*

Explanation: This message identifies the current DSNUTILB intercept worklist-management operation that is being performed by the started task.

User response: No action is required.

ABPS5102I ******message_continuation_number****** **(***DSNUTILB_statement_sequence_no.***) Event:** *DSNUTILB_event* **Status:** *DSNUTILB_event_status*

Explanation: The DSNUTILB worklist has been updated with the information that is presented in this message.

User response: No action is required.

ABPS5103I ***<***message_continuation_number***>* (<***DSNUTILB_statement_sequence_no.***>) event=<***DSNUTILB_event***>, status=<***DSNUTILB_event_status***>, return code=<***DSNUTILB_event_rc***>.**

Explanation: The DSNUTILB worklist has been updated with the information that is presented in this message.

User response: No action is required.

ABPS5104E ******message_continuation_number****** **Unable to save worklist due to duplicate utility ID.**

Explanation: The DSNUTILB worklist could not be saved because a worklist that has the same DSNUTILB utility ID has already been saved. Worklists cannot have duplicate utility IDs.

User response: No action is required.

ABPS5110I **DSNUTILB intercept operation was successful.**

Explanation: The current DSNUTILB intercept operation completed successfully.

User response: No action is required.

ABPS5111E ******message_continuation_number****** **DSNUTILB intercept operation failed**

Explanation: The current DSNUTILB intercept operation failed.

User response: To determine the cause of this failure, check any SQL errors that were reported in the log prior to this error.

ABPS5112W **TCB: <***tcb_address***> No worklist data found to delete for UTILID:** *db2_utility_id*

Explanation: The DB2 Utilities Enhancement Tool maintenance utility found no worklist data for the specified DB2 utility ID.

User response: No action is required.

ABPS5113I ******message_continuation_number****** **Worklist is in use by another utility. Owning Session: <***session_token***>**

Explanation: The worklist is in use by another utility at this time. DB2 Utilities Enhancement Tool will not intercept the DB2 utility execution because a worklist for that utility ID already exists and is currently in use by another utility job. This message provides the session token value of the owning utility session. See the preceding ABPS0101I message that contains a matching session token value to determine the job name and job ID of the utility job that is currently using the worklist.

User response: You can perform any of the following actions, as appropriate: change the utility ID in the DSNUTILB utility job that you want to intercept, wait until the job that is currently using the worklist completes, or (if the other utility terminated abnormally without ending its owning session) use the TERMINATE SESSION console command to terminate the owning session.

ABPS5550E LE preinitialization service failed. Operation=<*operation_name*>, RC=<*return_code*>.

Explanation: The specified Language Environment® (LE) preinitialization service operation failed with the specified return code.

User response: Contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPS9999S Message formatter failed. Message=<*message_id*>, RC=<*return_code*>, Reason=<*reason_text*>.

Explanation: An error occurred while formatting the specified message. If this error is related to obtaining or releasing storage, the message ABPS0802E or ABPS0803E is also issued to provide storage details.

User response: To determine the cause of the error, review the return code and reason text in this message. If you need assistance, contact IBM Software Support.

ABPU5001I DB2 Utilities Enhancement Tool <*product_version*>, FMID=<*product_fmid*>, COMP_ID=<*product_compid*>.

Explanation: This message provides the following information for your configuration: product name, version and release, FMID, and component ID.

User response: No action is required.

ABPU5002I Initialization is complete.

Explanation: The initialization processing for the DSNUTILB intercept component of the DB2 Utilities Enhancement Tool completed successfully.

User response: No action is required.

ABPU5003I Intercept completed.

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has completed intercept processing for this DB2 utility execution.

User response: No action is required.

ABPU5004I Analysis started. Step=*step_number*

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has started the analysis phase for this DB2 utility command.

User response: No action is required.

ABPU5005I Analysis completed. RC=<*return_code*>.

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has completed the analysis phase for this DB2 utility command.

User response: No action is required.

ABPU5006I Thread cancel started. Step=*step_number*

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has started the thread-cancel processing phase for this DB2 utility command.

User response: No action is required.

ABPU5007I Thread cancel completed. RC=<*return_code*>.

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has completed the thread-cancel processing phase for this DB2 utility command.

User response: No action is required.

ABPU5008I Utility execution started. Step=*step_number*

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has started the DB2 utility execution phase for this utility command.

User response: No action is required.

ABPU5009I Utility execution completed. SYS=<*system_abend_code*>, USR=<*dsnutilb_return_code*>.

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has completed the utility execution phase for the DB2 utility command. This message provides the return code from the DSNUTILB program (the USR value). If the DSNUTILB program terminated abnormally with a system abend, the message also provides the system abend code (the SYS value).

User response: No action is required.

ABPU5010I Allow threads started. Step=*step_number*

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has started the allow-threads processing phase for this DB2 utility command.

User response: No action is required.

ABPU5011I Allow threads completed. RC=*return_code*

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept has completed the allow-threads processing phase for this DB2 utility command.

User response: No action is required.

ABPU5012I Connected to started task ABPID=*product_identifier***.**

Explanation: The DB2 DSNUTILB job has connected to the specified DB2 Utilities Enhancement Tool started task.

User response: No action is required.

ABPU5013E Unable to connect to DB2 subsystem=*db2_ssid*

Explanation: The DB2 DSNUTILB job could not connect to the specified DB2 subsystem through the DB2 Utilities Enhancement Tool started task.

User response: Make sure that the required DB2 subsystem is operational.

ABPU5014I Delete blocker ID processing started. Step=*step_number*

Explanation: The DSNUTILB intercept component of the DB2 Utilities Enhancement Tool has started the delete-blocker-ID phase of thread blocker processing for the DB2 utility command. This message provides the step number of the DELETE_BLOCKER_ID step.

User response: No action is required.

ABPU5015I Delete blocker ID processing completed. RC=*return_code*

Explanation: The DSNUTILB intercept component of the DB2 Utilities Enhancement Tool has completed the DELETE_BLOCKER_ID step of thread blocker processing for the DB2 utility command. This step completed with the specified return code.

User response: No action is required.

ABPU5016E Utility abended. SYS=*system_abend_code***, USR=***dsnutilb_return_code*

Explanation: The DB2 Utilities Enhancement Tool DSNUTILB intercept was not able to complete the execution phase for the DB2 utility command because the DSNUTILB program terminated abnormally with a system abend. This message provides the system abend code (the SYS value) and the DSNUTILB return code (the USR value). The message is issued as a WTO message.

User response: To determine the cause of the error, look up the system abend code and the DSNUTILB return code in the appropriate IBM documentation.

ABPU5017E SORT EXIT ERROR: *error_reason*

Explanation: The DB2 Utilities Enhancement Tool detected an error in a sort exit that it uses for implementing the additional options for the DB2 LOAD utility. See the error reason that is specified in this message for an explanation of the error.

User response: If the error is related to a data conversion failure, correct the data and run the LOAD utility again. If the error is related to a product internal error, contact IBM Software Support.

ABPU5018I SORT execution completed. SYS=*system_abend_code***, USR=***dsnutilb_return_code*

Explanation: SORT has completed. This message provides the return code from the SORT program (the USR value). If the SORT program terminated abnormally with a system abend, the message also provides the system abend code (the SYS value).

User response: No action is required.

ABPU5019E SORT ended abnormally. DSNUTILB will be canceled with an S222 abend.

Explanation: SORT processing during DSNUTILB interception ended abnormally. The DSNUTILB program will be canceled with an S222 abend.

User response: Review the messages that were produced by the SORT program to determine the cause of the SORT failure. Then correct this problem and rerun the job. You can safely ignore the S222 abend in the DSNUTILB program.

ABPU5020E Cancel syntax member *cancel_syntax_member* **was not found.**

Explanation: The ABPBMAIN cancel syntax member specified in the ABPBMAIN_CANCEL_MEMBER of the OPTIONS was not found in the parameters library.

User response: Make sure that the required member exists in the parameters library and is correctly specified in the options member.

ABPU5021E Global syntax member *global_syntax_member* **was not found.**

Explanation: ABPBMAIN global syntax member specified in the ABPBMAIN_GLOBAL_MEMBER of the OPTIONS was not found in the parameters library.

User response: Make sure that the required member exists in the parameters library and is correctly specified in the options member.

ABPU5022E Subtask *module_name* terminated unexpectedly. SYS=*system_abend_code*, USR=*return_code*.

Explanation: A task that is attached by DSNUTILB interception services ended unexpectedly. If the program terminated abnormally with a system abend, the message provides the system abend code (the SYS value). The message provides the return code from the program (the USR value).

User response: Run the job again. If the problem persists, contact IBM Software Support.

ABPU5023E Unsupported SYSREC data set type RECFM=*record_format*, DS SEQNO=*data_set_sequence_number*.

Explanation: The product encountered a SYSREC data set with an unsupported record format. The SYSREC data set must have a RECFM of F or V. Spanned record formats and RECFM=U and D are not supported.

User response: Run the job again with an appropriate SYSREC data set.

ABPU5025E Generation of identity column values is not supported. Col: *column_name*.

Explanation: The table contains an identity column that is defined as GENERATE ALWAYS, or for which no field specification was provided. The product cannot generate values for identity columns.

User response: If the identity column is defined as GENERATE BY DEFAULT, consider providing a field specification for the column. If the identity column is defined as GENERATE ALWAYS, the product cannot be used to load the table.

ABPU5034E DATA TYPE <*data_type*> COLUMN <*column_name*> not compatible with SQLTYPE <*sql_type*>, COLNO <*column_number*>.

Explanation: The data type of the specified target column is not compatible with the data type of the source table column.

User response: Correct the definition of the target DB2 table and rerun the job.

ABPU5037E MSGTEXT <*message_text*>.

Explanation: The client returned the error messages after a failed RECV call.

User response: See the product job log for additional information about the error.

ABPU5048E ABPPIPE error: Function=x'10' LOOKUP_RC=<*return_code*>.

Explanation: The product intercepted an UNLOAD that was running in the Workload Manager (WLM). The batch job that triggered the UNLOAD requested that the UNLOAD terminate with errors. This event indicates that the batch job encountered errors during the load process. This message is only issued in the DSNUTILU WLM address space.

User response: See the batch job output for more information.

ABPU5051E The target table is not defined in DB2.

Explanation: The table that is the target of the load is not defined to DB2.

User response: Correct the table name or create the table in DB2 and then rerun the job.

ABPU5200E API Initialization failed.

Explanation: The DSNUTILB interface program failed to complete initialization. This failure occurred during initialization of the internal API.

User response: To determine the cause of the failure, review the messages in the job output that precede this message. Then correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPU5300I Processing will not be performed.

Explanation: No DSNUTILB intercept processing will occur for this DB2 utility execution.

User response: See the messages that precede this one to determine the reason for the interception failure. If you still want to perform DSNUTILB interception, correct any problems that the prior messages identify and then rerun the job.

ABPU5301I Thread cancel prevented by policy.

Explanation: Threads will not be blocked and canceled prior to running this DSNUTILB utility based on the intercept policy that is in effect.

User response: If you want to block and cancel threads for the utility, edit the intercept policy to provide this function and then restart the DB2 Utilities Enhancement Tool started task.

ABPU5302E Unable to rename DSNUTILB DD statements.

Explanation: This DSNUTILB utility execution will not be intercepted because DB2 Utilities Enhancement Tool could not rename the DSNUTILB DD statements. Existing DDNAMEs in the TIOT conflicted with all

available DDNAME renaming patterns.

User response: If possible, remove any DD allocations from the DSNUTILB job step that conflict with any of the following patterns: ABP$____, ABP#____, ABP@____, $ABP____, #ABP____, and @ABP____. If the conflicting DDNAME allocations cannot be removed, contact IBM Software Support for assistance.

ABPU5303E DDNAME rename operation failed for DDNAME=*original DD name*, new DDNAME=*new DD name*.

Explanation: This DSNUTILB utility execution will not be intercepted because DB2 Utilities Enhancement Tool could not rename the DSNUTILB DD statements.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5304E SWAREQ failed for DDNAME=*dd_name*, RC=*return_code*.

Explanation: The SWAREQ service returned a non-zero return code when it was called to provide the JFCB address for the specified DD name.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5305E DSNUTILB returned an error parsing the SYSIN data set.

Explanation: This DSNUTILB utility execution will not be intercepted because the DSNUTILB parser returned an error while parsing the SYSIN data set.

User response: See the error messages that were returned by DSNUTILB. Then correct the errors in the SYSIN data set and rerun the job.

ABPU5306E DSNUTILB syntax parser returned an error while parsing the SYSIN data set.

Explanation: This DSNUTILB utility execution will not be intercepted because the parser for the DB2 Utilities Enhancement Tool DSNUTILB statement returned an error while parsing the SYSIN data set.

User response: See the error messages that DSNUTILB returned. Then correct the errors in the SYSIN data set and rerun the job.

ABPU5307E Unable to determine restart status.

Explanation: This DSNUTILB utility execution will not be intercepted because DB2 Utilities Enhancement Tool could not determine the restart status for the utility ID.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5308I UTILID in use by stopped utility but no worklist exists.

Explanation: This DSNUTILB utility execution will not be intercepted because the utility ID is in use by a stopped utility and no worklist exists in the DB2 Utilities Enhancement Tool restart tables.

User response: No action is required.

ABPU5309I Move worklist failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted for the following reasons: a worklist for the specified utility ID already exists; no restartable utility was found; and the worklist move operation failed.

User response: Manually delete the worklist, as described in the user's guide, then rerun the job.

ABPU5310I Restart was specified but no stopped utility was found for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because a restart was requested and no stopped utility was found for this utility ID.

User response: Remove the restart parameter from the utility job, and then rerun the job.

ABPU5311E Save worklist failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because the worklist that is required for interception processing could not be saved.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5312E A running utility was found with utility ID=*utility_ID*.

Explanation: This DSNUTILB utility job will not be intercepted because another utility is already running with the same utility ID.

User response: Wait for the utility that is running to terminate, or specify a different utility ID for this utility job and rerun this job.

ABPU5313E Get next worklist step failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because the next step in the worklist that is required for interception processing cannot be retrieved.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5314E Update worklist status failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because the worklist status cannot be updated.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5315E Phase 2 policy processing failed.

Explanation: This DSNUTILB utility execution will not be intercepted because phase two of intercept policy processing failed.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5316E SET worklist step failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because the SET worklist step operation failed.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5317S Unable to locate USTI for current step UTILID=*utility_ID*, STEP=*utility_step*.

Explanation: This DSNUTILB utility execution will not be intercepted because an internal error occurred.

User response: Contact IBM Software Support.

ABPU5318E LISTDEF expansion failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because the LISTDEF that is specified for the utility ID cannot be expanded to determine the DB2 objects to process.

User response: Contact IBM Software Support.

ABPU5319E Save object list failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution will not be intercepted because the object list cannot be saved.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5320E SAPI processing failed, RC=*return_code*, RSN=*reason_code*.

Explanation: The SAPI processing service returned a non-zero return code while attempting to perform a SAPI function.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5321E SAPI processing failed, RC=*return_code*, RSN=*reason_code*.

Explanation: The SAPI processing component returned a non-zero return code while setting up a SAPI function.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5322E Listdef processing failed, RC=*return_code*, RSN=*reason_code*.

Explanation: The processing of the LISTDEF for the intercepted DB2 utility failed with the specified non-zero return code.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5323S A usable temporary LISTDEF name could not be generated.

Explanation: A usable, temporary LISTDEF name could not be generated because all of the temporary LISTDEF names known to DB2 Utilities Enhancement Tool occurred in the SYSIN data set for the utility job.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5324E Merge worklist failed for utility ID=*utility_ID*.

Explanation: This DSNUTILB utility execution cannot continue because the worklist that DB2 Utilities Enhancement Tool generated for utility restart purposes cannot be merged with the original worklist.

User response: See the error messages that are related to this error in the log for the DB2 Utilities Enhancement Tool started task.

ABPU5325I Restart in progress for utility ID=*utility_ID*

Explanation: The specified DB2 utility execution has been restarted at the request of the user.

User response: No action is required.

ABPU5326E Open failed for DSN=*data_set_name*

Explanation: A failure occurred while DB2 Utilities Enhancement Tool was trying to open the specified data set. Additional messages provide diagnostic information about this problem.

User response: See the related messages to diagnose the problem. After you resolve the problem, rerun the utility.

ABPU5327E Open failed. Abend code = *systemCompletionCode***, reason =** *reasonCode*

Explanation: A failure occurred while DB2 Utilities Enhancement Tool was trying to open a data set. This message provides the completion code and reason code for this failure.

User response: Resolve the problem that is causing the error and then rerun the job.

ABPU5328E Open failed. RC=*return_code*

Explanation: A failure occurred while DB2 Utilities Enhancement Tool was trying to open a data set. This message provides the return code from the OPEN macro.

User response: Resolve the problem that is causing the error and then rerun the job.

ABPU5329W Member not found in data set DSN=*data_set_name*

Explanation: A failure occurred while DB2 Utilities Enhancement Tool was trying to open a member of the specified data set. The member was not found in the data set.

User response: Resolve the problem that is causing the error and then rerun the job.

ABPU5330I Original DSNUTILB syntax follows:

Explanation: This message introduces the original, unmodified DSNUTILB syntax that was submitted for the utility. This syntax is presented in the message ABPU5331I, which follows this one. DB2 Utilities Enhancement Tool modifies this syntax before passing it to the DSNUTILB program.

User response: No action is required.

ABPU5331I *dsnutilb_syntax*

Explanation: This message contains all or part of the original, unmodified DSNUTILB syntax that was submitted for the utility.

User response: No action is required.

ABPU5332I End of original DSNUTILB syntax listing.

Explanation: This message indicates the end of the original, unmodified DSNUTILB syntax that was submitted for this utility and that is presented in the preceding message ABPU5331I.

User response: No action is required.

ABPU5333E TEMPLATE data set name processing failed, RC=*return_code***, RSN=***reason_code***.**

Explanation: The processing of the TEMPLATE data set name failed with the specified non-zero return code because an error occurred.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5334E TEMPLATE expansion failed for utility ID=*utility_ID***.**

Explanation: This DSNUTILB utility execution will not be intercepted because the TEMPLATE referenced in the LOAD utility statement could not be expanded to determine the data set name for the LOAD utility input.

User response: Contact IBM Software Support.

ABPU5335E UFSP processing failed, RC=*return_code***, RSN=***reason_code***.**

Explanation: The UFSP processing component issued a return code greater than 4 while setting up a UFSP function. The failure might occur because the table does not exist in DB2, or because the module could not obtain necessary storage space.

User response: Contact IBM Software Support. Provide the Support representative with the full text of this message.

ABPU5336E An error was detected during DB2 catalog lookup of column column_name.

Explanation: The DB2 Utilities Enhancement Tool UFSP processing component returned a non-zero return code while looking up information in the DB2 catalog.

User response: Contact IBM Software Support. Provide the Support representative with the full text of this message and the SYSPRINT log of the DB2 Utilities Enhancement Tool started task.

ABPU5337E The UFSP component detected an index column with an unsupported data type.

Explanation: The DB2 Utilities Enhancement Tool UFSP processing component detected an index key column with a data type that is not supported by the PRESORT option for the DB2 LOAD utility. These

unsupported data types are: REAL, DOUBLE, FLOAT, DECFLOAT, DISTINCT, BLOB, CLOB, and DBCLOB. The PRESORT option does not sort the data in input records by index key if the index key contains a column with an unsupported data type.

User response: If you want to sort the data in the input records for the LOAD utility by index key, you must do so manually.

ABPU5338E Session has been terminated by the server.

Explanation: The DSNUTILB interception did not complete because the session was terminated by the server.

User response: Check with the system administrator to determine the reason for the termination of the DSNUTILB interception program.

ABPU5339E Session creation failed RC=*return_code***, RSN=***reason_code***, Reason=***description*

Explanation: DSNUTILB interception failed to complete initialization. The failure occurred during the creation of an interception session for the DB2 utility.

User response: To determine the cause of the failure, review the reason description in this message. Correct the problem and run the job again. If you need assistance, contact IBM Software Support.

ABPU5340E Worklist in use by another utility ID=*utility_ID*

Explanation: DB2 Utilities Enhancement Tool will not intercept a DB2 utility execution that is associated with the specified utility ID because a worklist for that utility ID already exists and is currently in use by another utility job. See the corresponding message ABPS5113I in the SYSPRINT data set for the started task to determine the session token of the owning utility session.

User response: Either change the utility ID in the utility job that you want to intercept, or wait until the utility job that is currently using the worklist completes. Then rerun the utility job that failed to be intercepted.

ABPU5341E Unable to determine restart status for utility ID=*utility_ID*

Explanation: DB2 Utilities Enhancement Tool cannot determine whether the DB2 utility should be restarted because the status of the last utility operation within the current worklist step was not recorded in the intercept worklist tables. This situation might be caused by an unexpected system outage.

User response: Use the ABPMAINT utility to set the restart status for the utility. Specify one of these options for the utility: MARK_COMPLETE if the last utility operation completed successfully and the utility needs to be restarted from the next operation in the current worklist step, or FORCE_RESTART if the last utility operation needs to be restarted to complete its processing.

ABPU5342I -TERM UTILITY issued by user, cleaning up utility ID=*utility_ID*

Explanation: The -TERM UTILITY command was issued for the specified utility ID after the utility ended in a restartable state. The utility will complete its current worklist step and then terminate. Also, DB2 Utilities Enhancement Tool will automatically delete the data that is associated with this utility ID from the intercept worklist tables. The utility will no longer be restartable.

User response: No action is required.

ABPU5343I -TERM UTILITY issued during utility execution for utility ID=*utility_ID***.**

Explanation: The -TERM UTILITY command was issued for the specified utility ID while the utility was running. The utility will complete its current worklist step and then terminate. Also, DB2 Utilities Enhancement Tool will automatically delete the data that is associated with this utility ID from the intercept worklist tables.

User response: No action is required.

ABPU5344E Get discard table ROWID failed for utility ID=*utility_ID*

Explanation: This DSNUTILB utility execution will not be intercepted because the discard table ROWID cannot be retrieved.

User response: See the error messages that are related to this error in the log for the product started task.

ABPU5345E Unable to dynamically allocate data set. DD name=*ddname***, RC=***return_code***, RSN=***reason_code***.**

Explanation: The specified DD was not able to dynamically allocate a data set that was needed.

User response: Review messages in the JES job log to determine the cause of the dynamic allocation failure. Resolve the problem that is causing the error and then rerun the job.

ABPU5346E RDJFCB failed for DDNAME=*ddname***, RC=***return_code***.**

Explanation: The RDJFCB service returned a non-zero return code when it was called for the specified DD name.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5347E Open failed in ROUTINE=*routine* for DD name=*ddname*, RC=*return_code*.

Explanation: A failure occurred while the product was trying to open the specified DD name. This message provides the return code from the OPEN macro.

User response: Resolve the problem that is causing the error and then rerun the job.

ABPU5348E ATTACH failed for PROGRAM=*program_name*, RC=*return_code*.

Explanation: The ATTACH service returned a non-zero return code.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5349E IDENTIFY failed, RC=*return_code*.

Explanation: The IDENTIFY service returned a non-zero return code.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5350E The GET_SYSTEM_INFO call failed, RC=*return_code*.

Explanation: This DSNUTILB utility execution will not be intercepted because the system information could not be retrieved from the started task.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5351S I/O Hook installation failed because no matching DB2I was found.

Explanation: A severe internal error prevents DSNUTILB interception from continuing because the I/O hook cannot be successfully installed.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5352S I/O Hook installation failed, RC=*return_code*.

Explanation: A severe internal error prevents DSNUTILB interception from continuing because the I/O hook cannot be successfully installed.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5353S I/O Hook removal failed, RC=*return_code*.

Explanation: A severe internal error prevents DSNUTILB interception from continuing because the I/O hook cannot be successfully removed.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5354S Unknown UOBJ type encountered, UOBJ_OBJECT_TYPE=*uobj_object_type*.

Explanation: A severe internal error prevents DSNUTILB interception from continuing because the UOBJ object type is unknown.

User response: Contact IBM Software Support. Provide Support with the full text of this message.

ABPU5355E SYSREC <*sysrec_name*> was specified but no TEMPLATE or job step allocation was found.

Explanation: The utility statement syntax specified SYSREC but the name did not match any TEMPLATE or job step allocation.

User response: Provide a TEMPLATE or DD statement that matches the SYSREC name specified in the utility syntax.

ABPU5356W DSNUTILB syntax parser detected an empty SYSIN data set.

Explanation: This DSNUTILB utility execution will not be intercepted because the parser for the product detected an empty SYSIN data set.

User response: Correct the errors in the SYSIN data set and rerun the utility job.

ABPU5357E Tape data set detected for DDNAME=*ddname*

Explanation: DSNUTILB utility execution will not be intercepted because DB2 Utilities Enhancement Tool detected that the DDNAME represents a tape data set.

User response: No action is required.

ABPU5358E Column type is not supported for conversion to DB2 internal format. TYPE=<*columnType*>.

Explanation: The process for converting data to DB2 internal format does not support the specified column data type.

User response: Use the standard LOAD for the table, or change the data type of the column.

ABPU5359E Unable to dynamically allocate SYSREC dataset. RC= *return_code* **RSN=** *reason_code***.**

Explanation: A SYSREC data set could not be dynamically allocated. See message ABPU5360E for the data set name.

User response: Review messages in the JES job log to determine the cause of the dynamic allocation failure. Resolve the problem that is causing the error and then rerun the job.

ABPU5360E DSN=*data_set_name***.**

Explanation: The named data set could not be dynamically allocated. See message ABPU5359E for the dynamic allocation return and reason codes.

User response: Review messages in the JES job log to determine the cause of the dynamic allocation failure. Resolve the problem and then rerun the job.

ABPU5361E DEFAULTIF is not supported for partitioning key column *column_name***.**

Explanation: This DSNUTILB utility execution will not be intercepted because the product detected that the DEFAULTIF keyword is used with a column that participates in the partitioning key of the table. The DEFAULTIF keyword cannot be used with partitioning key columns.

User response: Correct the syntax and resubmit the job.

ABPU5363E Field *column_name* **not found.**

Explanation: During processing of the LOAD specifications, the product detected the specified column, which does not exist in the catalog and is not used for NULLIF or DEFAULTIF conditions. Because IGNOREFIELDS NO was specified, processing of the LOAD statement was terminated.

User response: Correct the LOAD utility syntax and run the job again.

ABPU5364I Record <*record_nbr***> discarded due to WHEN clause specification.**

Explanation: The record was discarded because it did not satisfy any of the WHEN clause conditions that are specified in the LOAD control cards.

User response: No action is required.

ABPU5400E Utility processing failed by policy practice *practice_name***.**

Explanation: The utility job step was terminated because the DB2 Utilities Enhancement Tool policy specified a fail return code.

User response: Correct the utility statement and rerun the job.

ABPU5401E Syntax denied: id=*string***.**

Explanation: The specified utility syntax is denied by the policy.

User response: Correct the utility statement and rerun the job.

ABPU5402E Syntax required: id=*string***.**

Explanation: The specified utility syntax is required by the policy.

User response: Correct the utility statement and rerun the job.

ABPU5403I Utility statement altered by policy practice *practice_name***.**

Explanation: The utility statement syntax was dynamically changed before utility execution in accordance with the specifications in the DB2 Utilities Enhancement Tool intercept policy.

User response: No action is required.

ABPU5404E Utility monitor encountered an error RC= *return_code* **RSN=** *reason_code***.**

Explanation: The utility monitor encountered an error while checking for syntax modifications.

User response: Contact IBM Software Support.

ABPU5405I Utility return code altered by policy practice *practice_name***.**

Explanation: The utility return code was changed by policy practice *practice_name*.

User response: No action is required.

ABPU5406E SQL function *sql_function* **failed with SQLCODE=** *sql_code*

Explanation: The started task encountered an error while executing a SQL function on behalf of the client.

User response: IBM Software Support

ABPU5407I SQL CREATE successful for mapping table *mapping_table_name*

Explanation: The product successfully created a mapping table and mapping table index for use by the REORG TABLESPACE utility.

User response: No action is required.

ABPU5408I SQL DROP successful for mapping table *mapping_table_name*

Explanation: The product successfully dropped a mapping table and mapping table index for use by the REORG TABLESPACE utility.

User response: No action is required.

ABPU5409I SQL CREATE successful for discard table *discard_table_name*

Explanation: The product successfully created a discard table space and a discard table for use by the CHECK DATA utility.

User response: No action is required.

ABPU5410I SQL DROP successful for discard table space <*discard_table_space_name***>.**

Explanation: The product successfully dropped a discard table space and, as a result, the associated discard table used by the CHECK DATA utility. Any authorizations granted to the <*authid*> running the utility are also automatically revoked by the table space drop.

User response: No action is required.

ABPU5411I GRANT INSERT successful to discard table for authid *db2_authid*

Explanation: The product successfully granted insert authority to the discard table used by the CHECK DATA utility.

User response: No action is required.

ABPU5412W SYSREC records discarded during CONVERT_INTERNAL processing. Utility return code altered.

Explanation: The utility return code was dynamically changed after utility execution because CONVERT_INTERNAL processing discarded one or more SYSREC records. SYSREC records may be discarded due to data validation or conversion errors or because records were found that did not belong to any partition that was included in the LOAD job.

User response: Correct the problem records in the SYSREC data set and rerun the job.

ABPU5500I Load pre-processing started.

Explanation: Syntax IFDISCARDS or SHRLEVEL REFERENCE was found in the load job input stream. Shadow objects will be created and loaded.

User response: No action is required.

ABPU5501I Load pre-processing finished with RC=*return_code***.**

Explanation: Preliminary actions for IFDISCARDS or SHRLEVEL REFERENCE finished with the specified return code.

User response: No action is required.

ABPU5502I Load post-processing started.

Explanation: The main load processing phase is complete, and additional actions will be performed for IFDISCARDS or SHRLEVEL REFERENCE processing.

User response: No action is required.

ABPU5503I Load post-processing finished with RC=*return_code***.**

Explanation: Additional actions for IFDISCARDS or SHRLEVEL REFERENCE were performed.

User response: No action is required.

ABPU5504E Storage release failed. Module=*module_name***, storage area=***storage_area_name***, RC=***return_code***.**

Explanation: The specified module failed while attempting to free the specified storage area.

User response: No action is required.

ABPU5505E Attempt to obtain storage failed. Module=*module_name***, storage area=***storage_area_name***, RC=***return_code***.**

Explanation: The specified module failed while attempting to obtain the specified storage area.

User response: Increase the region size that is available to the DB2 Utilities Enhancement Tool and run the job again. If the problem persists, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5506E SQL error.

Explanation: An SQL error occurred in the started task during load processing for the IFDISCARDS option or the SHRLEVEL REFERENCE option. Message ABPU5507I contains the error text.

User response: See *DB2 for Z/OS Messages* documentation for information about the DB2 messages that are displayed in ABPU5507I. If you are unable to resolve the error, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5507I ERRORTXT=*error_text*.

Explanation: SQL error diagnostic information.

User response: No action is required.

ABPU5508E Insufficient authority to load data into table space *table_space_name*.

Explanation: The user ID that submitted the job is not authorized to perform a load into the specified table space.

User response: Select another table space to load.

ABPU5509E Insufficient authority for load with STATS into table space *table_space_name.*

Explanation: When the STATISTICS keyword is specified in a load utility job, you must use a privilege set that includes the STATS privilege.

User response: Select another table space or remove the STATISTICS keyword.

ABPU5510E Operation *operation_name* on data set *data_set_name* failed, error number =*error_number_value*.

Explanation: The specified operation on the data set failed.

User response: See *z/OS UNIX System Services Messages and Codes* documentation for information about the displayed error. If you are unable to resolve the error, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5511E Data set operation failed. RC=*return_code.*

Explanation: The data set operation failed with the specified return code. Message ABPU5512I contains the error text.

User response: See *MVS System Messages* documentation for information about the messages that are displayed in ABPU5512I. If you are unable to resolve the error, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5512I *error_text.*

Explanation: Data set operation error text.

User response: No action is required.

ABPU5513E Compilation of regular expression failed. Expression=*expression_name*.

Explanation: The attempt to compile the specified regular expression failed. Message ABPU5515I contains the error text.

User response: Contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5514E Matching of regular expression failed. Expression: *expression_name*.

Explanation: The attempt to match the specified regular expression failed. Messages ABPU5515I and ABPU5516I contain the error text.

User response: Contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5515I ERRORTXT=*error_text*.

Explanation: An attempt to compile or match a regular expression pattern failed.

User response: No action is required.

ABPU5516I ERRORTXT=*error_text*.

Explanation: An attempt to match the regular expression input failed.

User response: No action is required.

ABPU5517E Dynamic allocation error. DDNAME=*dd_name*, operation=*operation_name*, RC=*return_code*.

Explanation: Dynamic allocation of the specified temporary DD for a DB2 utility failed with the specified return code.

User response: See *MVS Programming Authorized Assembler Services Guide for z/OS* documentation for information about the reported code. If you are unable to resolve the error, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5518E Invalid partition specified for table space *table_space_name*.

Explanation: The specified table space partition does not exist.

User response: Select another partition for the load job.

ABPU5519E Service function error. Service name=*service_name*, RC=*return_code*.

Explanation: The specified service function ended with a nonzero return code. If they are present, messages ABPU5520I and ABPU5521I contain the error text.

User response: See *DB2 for Z/OS Messages* documentation for information about the messages that are displayed in ABPU5520I and ABPU5521I. If you are unable to resolve the error, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5520I ERRORTXT=*error_text*.

Explanation: Failed service function input statements.

User response: No action is required.

ABPU5521I ERRORTXT=*error_text*.

Explanation: Failed service function output statements.

User response: No action is required.

ABPU5522E IFI error.

Explanation: An IFI error occurred in the started task during load processing for the IFDISCARDS option or the SHRLEVEL REFERENCE option. Message ABPU5523I contains the error text.

User response: See *DB2 for Z/OS: Codes* documentation for information about the messages that are displayed in ABPU5523I. If you are unable to resolve the error, contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5523I ERRORTXT=*error_text*.

Explanation: An IFI error occurred in the started task during load processing for the IFDISCARDS option or the SHRLEVEL REFERENCE option.

User response: No action is required.

ABPU5524I Some input records were discarded and IFDISCARDS PAUSE was specified.

Explanation: The load utility job paused with return code 4. The production table space was placed in read-only access mode (RO) and was not changed.

User response: Review the discarded records, and then restart or terminate the paused load job.

ABPU5525I Some input records were discarded, and IFDISCARDS FAIL was specified.

Explanation: The load utility job terminated with return code 8. The production table space was not changed.

User response: Review discarded records and correct the data for the load job.

ABPU5526I Utility was restarted after IFDISCARDS PAUSE. All valid records will be committed.

Explanation: The load utility job was restarted after IFDISCARDS PAUSE. All valid records will be committed.

User response: No action is required.

ABPU5527E Exception with RC=*return_code*.

Explanation: Load processing for the IFDISCARDS option or the SHRLEVEL REFERENCE option failed with the specified return code.

User response: Contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5528E Unexpected exception.

Explanation: A severe error occurred during load processing for the IFDISCARDS option or the SHRLEVEL REFERENCE option.

User response: Contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5529E Table space *table_space_name* has NPSI and was not loaded.

Explanation: The specified table space has a nonpartitioned secondary index (NPSI), which is not compatible with a partial load when the IFDISCARDS option or the SHRLEVEL REFERENCE option is specified.

User response: Load the entire table space (rather than partitions), or specify the INDEXDEFER option to instruct the product to ignore the NPSI.

ABPU5530E Table space *table_space_name* contains versioned rows and was not loaded.

Explanation: For the specified table space, the value of the OLDEST_VERSION column is less than the value of the CURRENT_VERSION column. Versioned objects are not supported when you specify the IFDISCARDS option or the SHRLEVEL REFERENCE option with RESUME YES.

User response: Reorganize the table space to ensure that the value of the OLDEST_VERSION column equals the value of the CURRENT_VERSION column, or specify RESUME NO.

ABPU5531E Table space *table_space_name* **has status** *space_status* **and cannot be loaded.**

Explanation: The specified table space is not in a supported access mode. The table space access mode must be read-write (RW), read-only (RO), or utility (UT).

User response: Start the object in RW, RO, or UT mode, and then submit the LOAD utility job again.

ABPU5532E Table space *table_space_name* **is VCAT-defined. VCAT-defined objects are not supported.**

Explanation: When the IFDISCARDS option or the SHRLEVEL REFERENCE option is specified, VCAT-defined table spaces are not supported.

User response: Select another table space to load.

ABPU5533E Table space *table_space_name* **contains an XML column. XML objects are not supported.**

Explanation: When the IFDISCARDS option or the SHRLEVEL REFERENCE option is specified, XML objects are not supported.

User response: Select another table space to load.

ABPU5534E Table space *table_space_name* **contains a LOB column. LOB objects are not supported.**

Explanation: When the IFDISCARDS option or the SHRLEVEL REFERENCE option is specified, LOB objects are not supported.

User response: Select another table space to load.

ABPU5535I DSCOPY_LIMIT value is *limit_value***.**

Explanation: The DSCOPY_LIMIT parameter specifies the maximum number of concurrent data set operations for load processing when the IFDISCARDS or SHRLEVEL REFERENCE option is specified. The default value is 0, which indicates that the product is to automatically determine the limit and display it in this message.

User response: In most cases, no action is required. However, if the load utility job abnormally ends due to insufficient memory, you can modify the DSCOPY_LIMIT value. Valid values are 0 - 250.

In member ABPDTDOP in data set *hlq.mlq*.SABPSAMP, specify a smaller value for DSCOPY_LIMIT than that displayed in this message, and then resubmit the load job.

ABPU5536I Load prevalidation restart handler started.

Explanation: A load utility job that specified the SHRLEVEL REFERENCE option or the IFDISCARDS option was restarted. Additional processing for shadow objects is required.

User response: No action is required.

ABPU5537I Load prevalidation restart handler finished with RC *return_code***.**

Explanation: A load utility job that specified the SHRLEVEL REFERENCE option or the IFDISCARDS option was restarted. Additional processing for shadow objects completed with the specified return code.

User response: No action is required.

ABPU5538I Table space *table_space_name* **is in check pending status.**

Explanation: The specified table space is involved in a referential relationship, and the load utility job contains the SHRLEVEL REFERENCE option or the IFDISCARDS option.

User response: No action is required.

ABPU5539I Table space *table_space_name* **is in check pending status.**

Explanation: The specified table space is involved in a referential relationship, and the parent table was loaded with the REPLACE option.

User response: No action is required.

ABPU5540I Index space *index_space_name* **is in rebuild pending status.**

Explanation: The specified index space contains a nonpartitioned secondary index (NPSI), and it is deferred with option INDEXDEFER.

User response: No action is required.

ABPU5541E Table space *table_space_name* **was altered with option ROTATE PARTITION.**

Explanation: When the IFDISCARDS option or the SHRLEVEL REFERENCE option is specified, table spaces with rotated partitions are not supported.

User response: Select another table space to load.

ABPU5542E Feature is not accessible in this version of DB2.

Explanation: The IFDISCARDS option and SHRLEVEL REFERENCE option require DB2 version 10 or later.

User response: Remove the unsupported option from the LOAD utility syntax.

ABPU5543E Index space *index_space_name* **has status** *space_status* **and its base table space cannot be loaded.**

Explanation: The specified index space is not in a supported access mode. The index space access mode must be read-write (RW), read-only (RO), or utility (UT).

User response: Start the object in RW, RO, or UT mode, and then submit the LOAD utility job again.

ABPU5544E *error_text***.**

Explanation: Keyword RESUME NO cannot be specified with the IFDISCARDS and SHRLEVEL REFERENCE options.

User response: Specify RESUME YES or REPLACE instead.

ABPU5545I Template data set was renamed.

Explanation: The product renamed the template data set.

User response: No action is required.

ABPU5546I Template name: *template_name***.**

Explanation: The product renamed the template data set as specified in the message text.

User response: No action is required.

ABPU5547I Old DSN: *old_data_set_name***.**

Explanation: The product renamed the template data set as specified in the message text.

User response: No action is required.

ABPU5548I New DSN: *new_data_set_name***.**

Explanation: The product renamed the template data set as specified in the message text.

User response: No action is required.

ABPU5549E Requested module *module_name* **not found.**

Explanation: The requested module was not found.

User response: Ensure that module *module_name* exists in the STEPLIB concatenation or the linklist.

ABPU5551I ERRORTXT *error_text***.**

Explanation: Service function failure explanation.

User response: No action is required.

ABPU5552I <*copy_output*>

Explanation: COPY command output is displayed.

User response: No action is required.

ABPU5553E Table space <*tablespace_name***> is a simple table space. Simple table spaces are not supported.**

Explanation: If a table space is neither segmented nor partitioned, and it is not a LOB table space, DB2 Utilities Enhancement Tool considers it a simple table space. Simple table spaces are not supported when you specify SHRLEVEL REFERENCE or IFDISCARDS.

User response: Specify another table space for the load job.

ABPU5556E Partition <*partition_number***> should be reorganized.**

Explanation: DB2 Utilities Enhancement Tool has detected that an alter has been performed on the specified partition since the last REORG utility was run. To make sure that the version is reset, run another REORG utility on the table space.

User response: Run a DB2 REORG utility on the specified partition before running the DB2 LOAD utility with the extended syntax.

ABPU5557E The required DB2 PTF <*ptf_number***> is not applied.**

Explanation: When image copies are requested to be taken during the LOAD process, the DB2 Utilities Enhancement Tool prevalidation feature (part of processing when you specify SHRLEVEL REFERENCE or IFDISCARDS), requires the following DB2 PTFs:

- DB2 10: UI30114
- DB2 11: UI30115

User response: Apply the required DB2 PTF for the version of DB2 that you are running.

ABPU5558E Table <*table_name*> was altered during the load process.

Explanation: Tables cannot be altered while performing LOAD SHRLEVEL REFERENCE or IFDISCARDS processing.

User response: Do not alter tables during load processing.

ABPU5559I Secondary LOAD utility with empty input file issued to create data sets for DEFINE NO.

Explanation: A secondary LOAD should be called for a DEFINE NO table space when you specify either the IFDISCARDS extended syntax option or the SHRLEVEL REFERENCE extended syntax option.

User response: No action is required.

ABPU5560I Secondary LOAD finished.

Explanation: A secondary LOAD should be called for a DEFINE NO table space when you specify either the IFDISCARDS extended syntax option or the SHRLEVEL REFERENCE extended syntax option.

User response: No action is required.

ABPU5701I Total records bypassed as outside partition selection: *discard_count*.

Explanation: SYSREC records were bypassed because they were not within the selected partition list.

User response: No action is required.

ABPU5730E DB2 call attachment facility error. RC=<*hex_return_code*> RSN=<*hex_reason_code*>.

Explanation: An attempt to connect to DB2 via the call attachment facility has failed. This error message can also indicate that the product was unable to load the call attachment facility into memory.

User response: Contact IBM Software Support.

**ABPU5732E Unable to LOAD MODULE <*module_name*>. RC=<*hex_return_code*> RSN=<*hex_reason_code.*

Explanation: An attempt to LOAD the named DB2 interface module failed. The system return code and reason code are also reported in the message.

User response: Verify that the DB2 load library is allocated in the STEPLIB or JOBLIB, and then resubmit the job. If you need further assistance, contact IBM Software Support.

ABPU5733I DOCTEXT <*xml_document_text*>.

Explanation: An attempt to parse the XML document reported in this message failed. Refer to messages ABPU5731E and ABPU5725I for more information on the stored procedure that returned the malformed document, and the parsing error.

User response: Contact IBM Software Support.

ABPU5800W Partition discovery failed in USE15. Record = *record_number*.

Explanation: Process USE15 could not determine the partition to which the record belongs. This is probably because the record is outside the range of the LIMITKEYS.

User response: Correct the partitioning key value in the identified SYSREC record and rerun the job. If you believe the record was erroneously discarded, contact IBM Software Support.

ABPU5801E Column <*column_name*> DEFAULT indicator value <*column_default_indicator*> is not supported.

Explanation: The product does not support the default indicator for SYSIBM.SYSCOLUMNS(DEFAULT) for this column.

User response: Supply data for this column or use a supported default type for the conversion to DB2 internal format, and then resubmit the job.

ABPU5802E Default value for column <*column_name*> is missing.

Explanation: When a column is defined as NOT NULL, you must provide a value or use the default value.

User response: Provide a valid value for the specified column and then resubmit the job.

ABPU5803E A failure has occurred in a data conversion routine.

Explanation: While trying to convert data, routine ABPUSE15 encountered an unrecoverable error.

User response: IBM Software Support Provide Support with all output from this job, including the dump.

ABPU5804E Unsupported column type. COLUMN <*column_name*> TYPE <*column_type*>.

Explanation: The data type for the specified column is not supported for the conversion to DB2 internal format.

User response: For information about the supported

data types, see the section about load processing enhancements in the product user's guide. Correct the error and then resubmit the job.

ABPU5805E Unsupported row format. FORMAT <*format_type*>.

Explanation: The table space row format is not supported when you are converting data to DB2 internal format, or the format is an unknown type. The supported row formats are basic and reordered. SYSIBM.SYSTABLEPART(FORMAT) shows the format type.

User response: Specify a supported format for the row and then resubmit the job.

ABPU5806E Column <*column_Name*> input data is too long.

Explanation: The input data that is specified for the column is longer than the length of the target column.

User response: Correct the LOAD or the table column definition, and then resubmit the job.

ABPU5806W Column *column_name* data is too long. Record = *record_number*.

Explanation: The input data is longer than the length of the target column.

User response: Correct the LOAD or the table column definition and resubmit the job.

ABPU5807W Column *column_name* has invalid data in record *record_number*.

Explanation: The data for the specified column is invalid.

User response: Correct the data in SYSREC and resubmit the job.

ABPU5809W DB2 size limit exceeded for column *column_name* record *record_number*.

Explanation: The value exceeds the DB2 size limits for the data type for the column.

User response: Correct the data in SYSREC and resubmit the job.

ABPU5810W Input numeric invalid column *column_name* record *record_number*.

Explanation: The input field contains an invalid numeric data type for the column.

User response: Correct the data in SYSREC and resubmit the job.

ABPU5812E IEAVPSE pause service failed. RC=*return_code*.

Explanation: The IEAVPSE pause release service failed with the specified return code.

User response: IBM Software Support Provide Support with the return code from this message.

ABPU5814E IEAVXFR transfer pause service failed, RC=*return_code*.

Explanation: The IEAVXFR transfer pause service failed with the specified return code.

User response: IBM Software Support Provide Support with the return code from this message.

ABPU5815E The SYSREC encoding scheme <*encoding_scheme*> does not match the table encoding scheme <*encoding_scheme*>.The INTO TABLE clause names a table with an unsupported ENCODING_SCHEME: <*encoding_scheme*>.

Explanation: The table encoding scheme must match the encoding scheme of the SYSREC data. The table encoding scheme is not compatible with the requested function. The product supports only the EBCDIC encoding scheme for the requested function.

User response: Either load the data to a table that has the same encoding scheme as the SYSREC data, or convert the SYSREC data to the encoding scheme that is used by the target table and run the load job again.Choose a table with the EBCDIC encoding scheme.

ABPU5817W Input packed decimal invalid for COLNAME <*column_name*>. RECORD <*record_nbr*>.

Explanation: The input field contains invalid packed decimal data for the column with type DECIMAL. Because no field specifications were provided, packed decimal data is expected.

User response: Correct the data in the SYSREC file and submit the job again.

ABPU5900E DB2 Utilities Enhancement Tool DSNUTILB exit module is not APF-authorized and is terminating.

Explanation: The load library for the DB2 Utilities Enhancement Tool DSNUTILB module is not APF-authorized, as required. Consequently, the DB2 Utilities Enhancement Tool DSNUTILB intercept processing for the DB2 utility is terminating.

User response: APF-authorize the load library for the DSNUTILB module, and then run DB2 utility job again.

ABPU5901E RVT locate operation failed

Explanation: DB2 Utilities Enhancement Tool could not locate its RVT control block.

User response: Make sure that at least one DB2 Utilities Enhancement Tool system is operational and then resubmit the job.

ABPU5901S RVT locate operation failed.

Explanation: DB2 Utilities Enhancement Tool could not locate its RVT control block.

User response: Make sure that at least one DB2 Utilities Enhancement Tool system is operational and then resubmit the job.

ABPU5902S COM locate operation failed.

Explanation: DB2 Utilities Enhancement Tool could not locate its COM control block.

User response: Make sure that at least one DB2 Utilities Enhancement Tool system is operational and then resubmit the job.

ABPU5903W DSNUTILF exit is inoperative for SSID: *db2_ssid***.**

Explanation: DSNUTILB intercept processing cannot be performed for the DB2 utility because the DB2 Utilities Enhancement Tool started task is not running or is not intercepting DSNUTILB for the specified DB2 subsystem ID (SSID). The DB2 utility job continues running.

User response: Make sure that at least one DB2 Utilities Enhancement Tool system is operational and enabled for interception. Also check for any additional messages that are related to the interception failure. After you correct any related errors and confirm that the system is ready for interception, resubmit the utility job.

If you receive reason code 0005, verify that the policy that is defined for the started task lists the correct subsystems. The group attach name is not a valid entry in the policy.

ABPU5904W DB2 Utilities Enhancement Tool is not active.

Explanation: DSNUTILB interception cannot be performed for the DB2 utility because the DB2 Utilities Enhancement Tool started task is not running. The utility continues running.

User response: Make sure that at least one DB2 Utilities Enhancement Tool system is operational and enabled for interception. Also, start the started task if necessary. Then resubmit the DB2 utility job.

ABPU5905W Load library open failed.

Explanation: DSNUTILB interception is currently unavailable. The utility continues running, but DSNUTILB interception will not occur.

User response: Make sure that a DB2 Utilities Enhancement Tool started task is operational. Also, make sure that an intercept policy is defined that allows interception for the DB2 subsystem on which you are running the utility job. If the problem persists, contact IBM Software Support.

ABPU5906W Load failed for ABPUMAIN.

Explanation: DSNUTILB interception is currently unavailable. The utility continues running, but DSNUTILB interception will not occur.

User response: Make sure that a DB2 Utilities Enhancement Tool started task is operational. Also, make sure that an intercept policy is defined that allows interception for the DB2 subsystem on which you are running the utility job. If the problem persists, contact IBM Software Support.

ABPU5907E SYSPRINT DD is missing or unusable.

Explanation: SYSPRINT DD is missing, or is allocated to DUMMY or NULLFILE.

User response: Supply a valid SYSPRINT DD statement in the JCL.

ABPU5908I IBM DB2 SORT found and will be used.

Explanation: IBM DB2 SORT was found and will be used for PRESORT on LOAD.

User response: No action is required.

ABPU5909W IBM DB2 SORT cannot be utilized. Not all modules found.

Explanation: Not all modules for IBM DB2 SORT were found.

User response: Ensure that IBM DB2 SORT has been installed correctly.

ABPU5910I DB2 Sort Program=*progname* **returned non-zero return code, RC=***rc*

Explanation: An internal error has occurred.

User response: Contact IBM Software Support. Provide the Support representative with the complete text of this message. Sorts will be performed by the default sort.

ABPU5911I DB2 sort program *program_name* abended. Default sort program will be used.

Explanation: The sort program abended. The default sort program will be used for sort processing.

User response: Contact IBM Software Support. Have available the listing that contains this message and any applicable related messages.

ABPU5912I ESTAE SDUMPX call RC=*return code*, RS=*reason code*.

Explanation: During **ESTAE** processing, a call to the z/OS **SDUMPX** facility returned the displayed return code and reason code.

User response: If RC=08, review the reason code in the appropriate SDUMPX documentation. Then make any changes to Dump Services that are needed to obtain proper diagnostic dumps. If you need assistance, contact IBM Software Support.

ABPU5913E LOAD PRESORT of hash table unable to proceed due to error.

Explanation: An error has occurred during **LOAD PRESORT** hash table analysis.

User response: Examine the job output and the DB2 Utilities Enhancement Tool Started Task to determine the cause of the error.

ABPU5914E Field length not supported for LOAD PRESORT: Column = *column_name*.

Explanation: The length of the data item specified for LOAD is not supported for PRESORT.

User response: Correct the length in the LOAD specification for the field in error.

ABPU5915E FORMAT DELIMITED is not supported for PRESORT with an ORGANIZE BY HASH table.

Explanation: **PRESORT** does not support SYSREC data that is in delimited file format where the target table is defined as ORGANIZE BY HASH.

User response: Provide a SYSREC that is not in delimited file format.

ABPU5916E Started task encountered an SQL error=*sql_code*

Explanation: An SQL error occurred.

User response: To determine the reason for the error, review the ABPS0202E messages that were issued in the started task address space, and see the DB2 messages documentation. If you need assistance, contact IBM Software Support.

ABPU5917E OPEN failed for SYSPRINT

Explanation: An OPEN macro failed for SYSPRINT.

User response: Review other messages issued to determine the cause.

ABPU5918E Field specification missing for a PRESORT key.

Explanation: A field specification is required for a field that is part of a PRESORT key.

User response: Provide a field specification for each field that is part of the PRESORT key.

ABPU9700I The output saved in the Autonomics Director history table exceeds 8M and is truncated.

Explanation: The 8-megabyte limit was reached for output in CLOB table in SYSAUTO.UTILITYRUNS_HISTORY. The product stops processing output.

User response: No action is required.

ABPU9701I Module BBY$NMIC not found in started task STEPLIB.

Explanation: Module BBY$NMIC was not found in the STEPLIB concatenation of the DB2 Utilities Enhancement Tool started task. DB2 Autonomics Director utility history collection is disabled.

User response: No action is required.

ABPU9702W Module BBY$NMIC does not conform to version 2, release 1 or later.

Explanation: Module BBY$NMIC found in the DB2 Utilities Enhancement Tool started task is not marked version 2 release 1 or later. DB2 Autonomics Director utility history collection is disabled.

User response: Ensure that you are using DB2 Utilities Solution pack version 2.1 or later.

ABPU9703W Module BBY$NMIC contains invalid offset to data.

Explanation: The module BBY$NMIC that was found in the DB2 Utilities Enhancement Tool started task contains an offset to the data structure that does not point to a valid version. DB2 Autonomics Director utility history collection is disabled.

User response: Contact IBM Software Support.

ABPU9704W BLDL error encountered searching for module BBY$NMIC. RSN=*reason_code*.

Explanation: The product encountered an error while searching for module BBY$NMIC. DB2 Autonomics Director utility history collection is disabled.

User response: Contact IBM Software Support.

ABPU9705W Error encountered attempting to load module BBY$NMIC.

Explanation: The product encountered an error while attempting to load module BBY$NMIC. DB2 Autonomics Director utility history collection is disabled.

User response: Contact IBM Software Support.

DB2 Utilities Enhancement Tool codes

Review the descriptions of the return codes and abend codes that can be issued from the DB2 Utilities Enhancement Tool started task, DSNUTILB intercept, or batch interface to determine how DB2 Utilities Enhancement Tool processing ended. For descriptions of any DB2 codes that might be returned for DB2 Utilities Enhancement Tool processing, see the *DB2 for z/OS Codes* manual.

Abend codes

The started task or any DB2 Utilities Enhancement Tool interface can issue a user abend code when it encounters an internal error. If you receive an abend code, contact IBM Software Support.

Return codes from the DSNUTILB intercept

For DSNUTILB intercept processing that you perform for DB2 utilities, DB2 Utilities Enhancement Tool issues a return code for the various processing phases that were performed for a utility command (a worklist step) and for the entire worklist step. You can find the return codes in the DB2 Utilities Enhancement Tool messages that are included in the SYSPRINT data set and JESMSGLG data set for the DB2 utility. (For thread-cancelation processing, the return codes in these messages might be from either the DSNUTILB intercept or the batch interface; the intercept uses the batch interface for intercept processing.) If a return code less than 8 is issued for a worklist step, the DSNUTILB intercept ignores the return code and continues processing. If a return code of 8 or higher is issued, intercept processing terminates.

The following table describes the return codes that the DSNUTILB intercept issues. For the return codes that the batch interface issues, see Table 33 on page 437.

Table 32. Return codes from the DSNUTILB intercept

Return Code	Description
00	A DSNUTILB intercept worklist step completed successfully.
04	Thread-cancelation processing completed without error in the following situations: • DB2 Utilities Enhancement Tool identified no active threads to cancel because either no active threads existed on the DB2 subsystem or none of the existing active threads matched the selection criteria in the intercept policy. • DB2 Utilities Enhancement Tool did not block new threads from forming on one or more of the DB2 objects. Because the ON_FAILURE (CONTINUE) parameter is specified in the *abpid*BGLB member (where *abpid* is the started task configuration ID), DB2 Utilities Enhancement Tool skipped thread blocking and continued to the next cancel request when a thread-cancelation failure occurred.

Table 32. Return codes from the DSNUTILB intercept (continued)

Return Code	Description
08	This utility command completed but DB2 Utilities Enhancement Tool could not confirm that all threads were canceled for that utility command. This return code is issued in the following situations: • After completing cancel processing of all threads, DB2 Utilities Enhancement Tool checked the status of the canceled threads to confirm that they were actually terminated. DB2 Utilities Enhancement Tool performed this checking up until the maximum number of checking retries was reached, as specified by the CHECK_THDTERM_RETRY_COUNT parameter. When this retry limit was reached, at least one thread was still active. Because no further checking was performed, DB2 Utilities Enhancement Tool could not confirm that all of the canceled threads eventually terminated. For more information, see the messages in the utility job output. • A failure occurred during the cancel processing. Because the ON_FAILURE (CONTINUE) parameter is specified in the *abpid*BGLB member, the utility command could continue to the next cancel request and eventually complete.
12	Because the CANCEL_TYPE parameter in the *abpid*BCAN member is set to NOBACKOUT, DB2 Utilities Enhancement Tool checked for outstanding units of recovery prior to canceling threads. At least one thread had an outstanding unit of recovery and therefore was not canceled.DB2 Utilities Enhancement Tool does not cancel threads that have an outstanding unit of recovery to prevent potential data integrity problems. Depending on whether you set the ON_FAILURE parameter in the *abpid*BGLB member to TERMINATE, the thread-cancelation processing either terminated or continued to the next thread that was identified by the rules in the intercept policy. By default, thread-cancelation processing continues.

Return codes from the batch interface

For thread-cancelation processing that you perform as part of batch jobs, DB2 Utilities Enhancement Tool issues a return code for the entire thread-cancelation job step and for each cancel request in the job step. You can find the return code for the entire job step in the JES message log (JESMSGLOG data set) for the batch job, and you can find the return code for a cancel request in the summary of cancel request processing (SPRT*nnnn*). If the job included multiple CANCEL_THREADS requests, the return code for the entire job step will be the highest return code that was received from any of the cancel requests.

The following table describes the return codes that the batch interface can return for batch thread-cancelation processing:

Table 33. Return codes from the batch interface

Return Code	Description
00	The job or cancel request completed successfully. All threads that matched the thread-filtering criteria were canceled.

Table 33. Return codes from the batch interface (continued)

Return Code	Description
04	The job or cancel request completed without error in the following situations: • DB2 Utilities Enhancement Tool identified no active threads to cancel because either no active threads existed on the specified DB2 subsystem or none of the existing active threads matched the CANCEL_THREADS parameters. • Thread-blocking was enabled for the job. However, DB2 Utilities Enhancement Tool failed to block new threads from forming on one or more of the specified DB2 objects. Because the ON_FAILURE (CONTINUE) parameter was specified, DB2 Utilities Enhancement Tool proceeded to cancel the existing, active threads on these objects and then continued to the next CANCEL_THREADS request. • Thread-blocking was enabled for the job. However, at least one of the CANCEL_THREADS requests in the job did not include any parameters for identifying the DB2 objects for which to block threads. Threads might have been canceled for this cancel request, but no thread blocking occurred.
08	This job completed but DB2 Utilities Enhancement Tool could not confirm that all threads were canceled. This return code is issued in the following situations: • After completing cancel processing for all threads, DB2 Utilities Enhancement Tool checked the status of the canceled threads to confirm that they were actually terminated. DB2 Utilities Enhancement Tool performed this checking up until the maximum number of checking retries was reached, as specified by the CHECK_THDTERM_RETRY_COUNT parameter. When this retry limit was reached, at least one thread was still active. Because no further checking was performed, DB2 Utilities Enhancement Tool could not confirm that all of the canceled threads eventually terminated. For more information, see the messages in the job output. • A failure occurred during the cancel processing of a CANCEL_THREADS request in the job. Because the ON_FAILURE (CONTINUE) parameter was specified, the job could continue to the next CANCEL_THREADS request and eventually complete.
12	The CANCEL_TYPE parameter was set to NOBACKOUT for a CANCEL_THREADS request. This option caused DB2 Utilities Enhancement Tool to check for outstanding units of recovery prior to canceling threads. DB2 Utilities Enhancement Tool found that a thread had an outstanding unit of recovery. To prevent potential data integrity problems, this thread was not canceled. DB2 Utilities Enhancement Tool will not cancel any threads that have an outstanding unit of recovery. Depending on whether you set the ON_FAILURE parameter to TERMINATE, the job either terminated or continued to the next thread that was identified by the CANCEL_THREADS request. By default, a job will continue and cancel the other threads.
16	Not currently used.
20	Not currently used.
24	Not currently used.
28	An error occurred while a thread-cancelation job was running. If you specified the ON_FAILURE (TERMINATE) parameter, DB2 Utilities Enhancement Tool will issue this return code when either a thread-blocking or thread-cancel failure occurs. To determine the cause of the failure, view the messages in the job output.
32	DB2 Utilities Enhancement Tool detected syntax errors in the ABPPARMS input control cards while parsing the control card parameters. For more information about the parameter errors, see the messages in the job output.

Related concepts:

"Return code use" on page 122
You can use the return codes that the DB2 UET batch interface returns for a thread-cancelation job step to control whether subsequent applications and programs in the batch job run.

"Determining whether thread blocking and canceling occurred" on page 234
DB2 UET generates several types of output for a thread-cancel or thread-blocker job step. Check this output to determine if your job step completed successfully.

Diagnostic information for Support

If you encounter a problem and need to contact IBM Software Support, you must gather certain information about your DB2 UET system and the problem before contacting Support. Your Support representative will need this information to correctly diagnose and resolve the problem.

Provide Support with the following types of diagnostic information:

- *(Required)* The DB2 UET version.
- *(Required)* The identifier for the latest DB2 UET APAR or PTF that has been applied on your system.
- *(Required)* The operating system type, version, and maintenance level.
- *(Required)* Your DB2 version and whether you are using DB2 data sharing.
- *(Required)* All output from the DB2 UET started task.
- (*DSNUTILB intercept users only*) Your DSNUTLB intercept policy and all output for the DB2 utility execution for which the problem occurred
- *(Batch interface users only)* All output from your DB2 UET batch job.
- *(ISPF interface users only)* A description of the activity that you were performing in the interface when the problem occurred, including a screen capture of the relevant ISPF panel, if possible. Also, provide the contents of the log for the TSO user who was using the interface.
- The complete contents of any dumps that Support requested. See "Producing dumps for diagnostic use."
- Any messages in the z/OS System Log that might pertain to the problem.

Your Support representative will provide instructions for transmitting this information.

Related tasks:

"Producing dumps for diagnostic use"
You might need to produce a dump of the started task address space or of multiple DB2 UET address spaces at the request of Support. The dump data can help Support diagnose a problem that you report.

Producing dumps for diagnostic use

You might need to produce a dump of the started task address space or of multiple DB2 UET address spaces at the request of Support. The dump data can help Support diagnose a problem that you report.

About this task

You should request only one dump at a time on your z/OS system.

To produce a system dump of the started task address space only, you can issue the following Modify operator command from the z/OS console:

```
F started_task_name,DUMP
```

Where *started_task_name* is the name of the DB2 UET started task configuration.

To produce a system dump of multiple address spaces (for example, a dump of the address spaces for the started task and the user interface that you are using), issue the MVS DUMP command from the z/OS console:

```
DUMP COMM=(dump_title)
R id,JOBNAME=(name1,name2,...),SDATA=(CSA,LPA,LSQA,PSA,RGN,SUM,SWA,TRT)
```

Where:

- *dump_title* is a name that you assign to the dump.
- *id* is the reply identification number, as specified in system message IEE094D.
- *name2,name2, ...* are values that identify the DB2 UET address spaces to dump. A name value can be:
  - The DB2 UET started task name (if you are dumping the started task address space)
  - A batch job name (if you are dumping a batch interface job or a DSNUTILB batch job)
  - The TSO user ID of an ISPF interface user (if you are dumping the ISPF interface address space)
- SDATA specifies the options that indicate the specific storage areas to dump.

For detailed information about the DUMP command, including descriptions of the SDATA options, see the IBM publication *MVS System Commands*.

What to do next

After you produce a dump, send it to Support. Your Support representative will provide transmittal instructions.

Chapter 13. Reference

These topics provide reference information about troubleshooting Tools Customizer and the DB2 UET customization and options module parameters, started task initialization options, z/OS console commands, user exits, dump commands, and diagnostic information for support.

Topics:

Tools Customizer reference

Before you use Tools Customizer, you should understand the Tools Customizer terminology and the data sets that Tools Customizer uses during customization.

Tools Customizer terminology

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

Products and components
How an IBM Tool is packaged determines whether it is referred to as a product or as a component in the Tools Customizer documentation and interface. An IBM Tool that is ordered as a stand-alone entity (that is, not as part of a solution pack) is referred to as a product. An IBM Tool that is part of a solution pack is referred to as a component. Some IBM Tools are available in both formats; therefore, the same IBM Tool can be referred to as a product or as a component depending on how it is packaged.

DB2 entry

You can customize DB2 UET on one or more DB2 entries. A DB2 entry can be any of the following items:

DB2 subsystem
A distinct instance of a relational database management system (RDBMS) that is not part of a data sharing group. An example of a DB2 subsystem name is DB01.

DB2 group attach name
The name that is used by the TSO/batch attachment, the call attachment facility (CAF), DL/I batch, utilities, and the Resource Recovery Services attachment facility (RRSAF) as a generic attachment name. An example of a group attach name is DSG1.

DB2 data sharing member
A DB2 subsystem that is assigned by the cross-system coupling facility (XCF) to a data sharing group. An example of a DB2 data sharing member name is DB02.

Tools Customizer maintains the following lists of DB2 entries:

Associated list

The list of DB2 entries that are associated with DB2 UET. If the product to be customized requires DB2 entries, you can customize DB2 UET only on DB2 entries that are in the associated list. When you customize DB2 UET, this list is displayed in the DB2 Entries, Associations, and Parameter Status section of the Customizer Workplace panel.

You can add and copy DB2 entries to the associated list. When you add or copy DB2 entries to the associated list, the entries are associated with DB2 UET.

Master list

The list of all DB2 entries that are defined but are not associated with DB2 UET. Tools Customizer obtains information about these DB2 entries either from entries that were created manually or from the customizations of other products that were discovered. If you remove a DB2 entry from the associated list, the DB2 entry is added to the master list. When you create a new DB2 entry, it is added to the master list, and when you associate the new entry with DB2 UET, it is removed from the master list and added to the associated list. The master list is displayed on the Associate a DB2 Entry for Product panel.

If the associated list does not have the DB2 entries on which you want to customize DB2 UET, you can associate existing entries from the master list to the associated list.

You can create new DB2 entries and copy existing entries to the master list.

High-level qualifier
The high-level qualifier is considered to be all of the qualifiers except the lowest level qualifier. A high-level qualifier includes a mid-level qualifier.

Product parameters
Parameters that are specific to DB2 UET. These parameters are defined by DB2 UET and are stored in a data member that is defined by DB2 UET.

DB2 parameters
Parameters for a DB2 entry. These parameters are defined by Tools Customizer and are stored in a DB2 parameter data member.

Status type

Product, LPAR, and DB2 entry status type

After you specify the product that you want to customize, the product, the LPAR, and the DB2 entries have a status. The status is partly based on whether required parameters are defined. For some products, LPAR parameters or DB2 parameters might not be required. In these cases, the status is Not Required.

To customize DB2 UET, all of the required parameters must be defined.

If required parameters for the the product parameters or DB2 parameters are not defined, the status of the parameters is Incomplete. Define values for parameters by manually editing them or by generating the customization jobs and specifying values for all of the required parameters that are displayed on the panels.

When values for all of the required parameters are defined, the status is Ready to Customize. Customization jobs can be generated only when all of the required parameters are defined and the status is Ready to Customize or Customized for the product parameters and DB2 parameters for the DB2 entries on which DB2 UET will be customized.

The following table shows the meaning of the status types. Each status is defined differently for each type of parameter.

Table 34. Status types for the product, the LPAR, and the DB2 entries

Status	Product	LPAR	DB2 entries
Incomplete	The required product parameters are not defined, or the required product parameters are defined but LPAR parameters, DB2 parameters, or both are not defined.	The required parameters are not defined.	The required parameters are not defined.
Discovered	The product parameter definitions were discovered by using the product Discover EXEC.	N/A	N/A
Ready to Customize	The required product, LPAR, and DB2 parameters are defined, the status is Ready to Customize or Customized for the LPAR and at least one associated DB2 entry. You can generate the customization jobs.	The required LPAR parameters are defined or LPAR parameters are not required.	The required DB2 parameters are defined or DB2 parameters are not required.
Customized	The jobs are customized on the local LPAR.	The jobs are customized for the product or for all of the associated DB2 entries on the local LPAR.	The jobs are customized for the DB2 entry.
Errors in Customization	N/A	N/A	Errors occurred while the customization jobs were being generated.
Not Required	N/A	LPAR parameters are not required.	DB2 parameters are not required.

Related tasks:

"Creating and associating DB2 entries" on page 78
You can create new DB2 entries and associate them with DB2 UET.

"Copying DB2 entries" on page 87
You can copy associated and not associated DB2 entries to other DB2 entries or to new DB2 entries.

"Removing DB2 entries" on page 89
You can remove DB2 entries from the associated list.

Data sets that Tools Customizer uses during customization

Tools Customizer uses several unique data sets during the customization process. Familiarize yourself with these data sets before you begin to use Tools Customizer.

Several different data sets are required to customize DB2 UET with Tools Customizer. These data sets are supplied by DB2 UET, supplied by Tools Customizer, or allocated by Tools Customizer.

DB2 UET provides the following data sets:

Metadata library
Contains the metadata for the product to be customized. Tools Customizer uses the metadata to determine which tasks, steps, and parameters to display on the Product Parameters panel, the LPAR Parameters panel, and the DB2 Parameters panel. This data set also contains the templates that Tools Customizer uses to generate the customization jobs.

The metadata library naming convention is *high_level_qualifier*.SABPDENU, where *high_level_qualifier* is all of the segments of the data set name except the lowest-level qualifier.

You specify the metadata library on the Specify the Metadata Library panel. READ access to this data set is required.

Discover EXEC library
Contains the DB2 UET Discover EXEC. When you customize DB2 UET, you can use the Discover EXEC to automatically retrieve and store product information, such as parameter values from an already customized product. Tools Customizer saves the discovered information in the data store.

The default name of the data set is the high-level qualifier for the metadata library plus a lowest-level qualifier. For DB2 UET, the lowest-level qualifier is SABPDENU. You can change the default value on the Discover Customized Product Information panel. EXECUTE access to this data set is required.

Tools Customizer provides the following data sets:

Tools Customizer metadata library
Contains the metadata for the DB2 and LPAR parameters that are required to customize DB2 UET. Tools Customizer uses the metadata to determine which parameters to display on the DB2 Parameters panel and the LPAR Parameters panel. In addition, Tools Customizer uses information in the metadata library to determine whether additional DB2 and LPAR parameters need to be displayed on these panels. As you customize different products, different DB2 and LPAR parameters might need to be defined.

The default name of the data set is DB2TOOL.CCQ110.SCCQDENU. You can change the default value on the Tools Customizer Settings panel. READ access to this data set is required.

Tools Customizer table library
Stores information about jobs that are customized. Job information that is

stored includes a description of the job, its member name and template name, the SSID, group attach name, and when the job was generated.

The default name of the data set is DB2TOOL.CCQ110.SCCQTENU. WRITE access to this data set is required.

Tools Customizer requires that the following data sets exist during the customization process. If the data sets do not exist, Tools Customizer automatically allocates them.

Discover output data set
Contains the output that is generated when you run the DB2 UET Discover EXEC. The DB2 UET Discover EXEC retrieves the metadata and values for the parameters from a previous customization of DB2 UET.

The default name of the data set is DB2TOOL.CCQ110.DISCOVER. You can change the default value on the Tools Customizer Settings panel or the Discover Customized Product Information panel. WRITE access to this data set is required.

Data store data set
Contains product, LPAR, and DB2 parameter values, and DB2 entry associations. Tools Customizer uses this data set to permanently store all information that is acquired about the product, DB2 subsystems or data sharing groups, and LPAR when you customize products on the local LPAR.

The default name of the data set is DB2TOOL.CCQ110.DATASTOR. You can change the default value on the Tools Customizer Settings panel. WRITE access to this data set is required.

Customization library
Contains the customization jobs that Tools Customizer generates for DB2 UET.

Tools Customizer checks whether a customization library name was specified for more than one instance of the same version of the same product. If the same customization library name is specified for more than one product of the same version, the CCQD123E message is issued to prevent you from overwriting previously generated customization jobs. Ensure that you specify unique qualifier for the customization library for each instance of the product.

To customize DB2 UET, submit the members of the data set in the order in which they are displayed on the Finish Product Customization panel.

The data set naming convention is *hlq*.$*LPAR_name*$.*xyzvrm*, where:

- *hlq* is the value of the **Customization library qualifier** field on the Tools Customizer Settings panel (CCQPSET)
- *LPAR_name* is the four-character LPAR name
- *xyzvrm* is the three-letter product identifier with the version, release, and modification level

For example, the data set name might be DB2TOOL.PRODUCT.CUST.$MVS1$.*XYZ*410.

WRITE access to this data set is required.

Tools Customizer allocates the data sets for the discover output, the data store, and the customization library with the attributes that are shown in the following table:

Table 35. Data set attributes for allocating the Discover output, data store, and customization library data sets

Data set	Organization	Record format	Record length	Block size	Data set name type
Discover output data set	PO	Variable block	16383	32760	LIBRARY
Data store data set	PO	Variable block	16383	32760	LIBRARY
Product customization library	PO	Fixed block	80	32720	LIBRARY

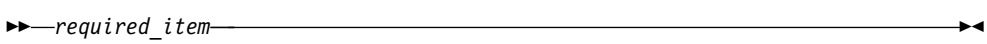
Restrictions:

- Multiple users cannot simultaneously share the discover output data set, data store data set, Tools Customizer metadata library, and metadata library.
- You cannot share the data store data set across multiple LPARs with shared DASD or copy the data store data set to another LPAR. Tools Customizer creates many cross-references between product and DB2 associations. Therefore, if you share or copy the data store data set, member names that are empty or that do not exist might be generated.

How to read syntax diagrams

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

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

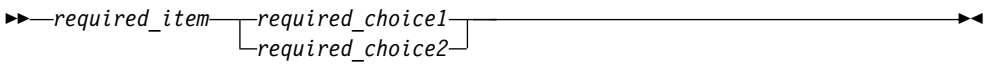


- Optional items appear below the main path.

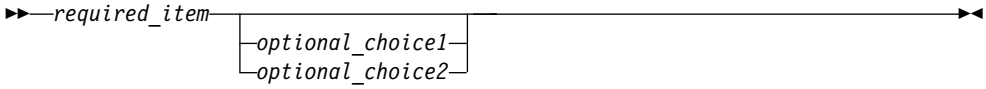
  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

- If you can choose from two or more items, they appear vertically, in a stack.

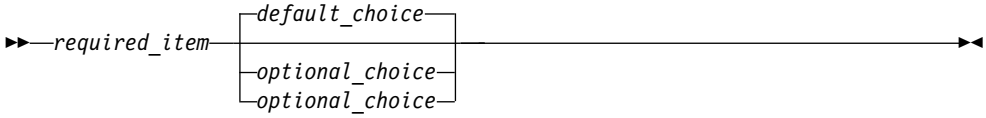
  If you *must* choose one of the items, one item of the stack appears on the main path.



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



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



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

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

A repeat arrow above a stack indicates that you can repeat the items in the stack.

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

Console commands for the started task

DB2 UET supports several z/OS console commands for the started task. You issue these commands by using the Modify operator command.

Syntax

The Modify command is `F` if issued from the z/OS console or `/F` if issued from SDSF.

Use the following syntax to issue a console command from the z/OS console:

```
F started_task_name,command_name
```

Use the following syntax to issue a console command from SDSF:

```
/F started_task_name,command_name
```

Where *started_task_name* is the name of the DB2 UET started task and *command_name* is the name of a supported console command. Separate the names with only a comma.

For some commands, you can optionally add an option such as GLOBAL after the command name. In this case, specify the command name, a comma, and then the option name (without blank spaces between these items), as follows:

```
F started_task_name,command_name,option
```

Commands

Tip: To list all DB2 UET console commands in the started task output, issue the **HELP** console command.

ACTIVATE INTERCEPT[,GLOBAL]
Use this command to enable DSNUTILB interception for the specified started task. You must have first defined a DSNUTILB intercept policy that identifies the DB2 subsystems and the utilities, users, or objects for which to perform intercept actions. These actions include implementing the supported LOAD and REORG TABLESPACE utility enhancements and blocking and canceling threads. You can optionally include the `GLOBAL` option. Use the `GLOBAL` option if you previously disabled DSNUTILB interception across the entire z/OS image by issuing the `DEACTIVATE INTERCEPT,GLOBAL` command and now want to re-enable DSNUTILB interception for the entire z/OS image.

DEACTIVATE INTERCEPT[,GLOBAL]
Use this command to disable DSNUTILB interception for the specified started task. You can optionally include the `GLOBAL` option to disable DSNUTILB interception across the entire z/OS image.

DISPLAY INTERCEPT[,GLOBAL|,ALL]
Use this command to write the local DSNUTILB interception status (Enabled or Disabled) for the specified started task to the SYSPRINT data set that is allocated to the started task. You control this status by issuing the `ACTIVATE INTERCEPT` or `DEACTIVATE INTERCEPT` command. You can optionally include the `GLOBAL` option to display the global interception status for the entire z/OS image. You control this status by issuing the `ACTIVATE INTERCEPT,GLOBAL` or `DEACTIVATE INTERCEPT,GLOBAL` command. Alternatively, you can include the `ALL` option to write all of the following information to the SYSPRINT data set:

- local interception status
- global interception status
- list of the DB2 SSIDs for which DSNUTILB interception is occurring, including the ABPID of each started task instance that is involved in interception processing

DISPLAY MEPL

Use this command to write a list of all DB2 UET modules to the SYSPRINT data set that is allocated to the started task. For each module, the list shows the module maintenance level, the date and time when the module was built, and other information for diagnostic use. Usually, you issue this command when directed to do so by Support.

DISPLAY POLICY

Use this command to write the contents of the DSNUTILB intercept policy for the specified started task to the SYSPRINT data set that is allocated to the started task. This information includes the DB2 subsystems and rules that are defined in your policy member (*abpid*PLCY). Note that the rules in the DISPLAY POLICY output are numbered. These numbers are referenced by the ABPRDIAG data set, which indicates, for each rule, whether interception occurred.

DISPLAY PRACTICE[,*ssid* | ,*practice_name*]

Use this command to generate a practice report that details the current practice, a specific practice, or all of the practices that are defined in the policy for the specified subsystem ID. The options that you specify control the information that is displayed, as follows:

- To display all of the defined practices, use the command without options.
- To display the current practice for the specified subsystem, use the command **DISPLAY PRACTICE**,*ssid*.
- To display the specified practice, use the command **DISPLAY PRACTICE**,,*practice_name*.
- To display the specified current practice, use the command **DISPLAY PRACTICE**,*ssid*,*practice_name*.

Issue the **MODIFY** command from SDSF as follows: `/F started_task_name,DISPLAY PRACTICE,,PRACTICE_1`

This command produces the following report in SYSPRINT for the specified practice (PRACTICE_1):

```
ABPS0814I 110 03:39:13.22 Command issued: DISPLAY PRACTICE,,PRACTICE_1

ABPS0836I 110 03:39:13.22 DB2 UTILITIES ENHANCEMENT TOOL practice info
ABPS0836I 110 03:39:13.22 ------------------------------------------
ABPS0836I 110 03:39:13.22  >   PRACTICE NAME="PRACTICE_1"
ABPS0836I 110 03:39:13.22  >     UTILITY NAME="LISTDEF"
ABPS0836I 110 03:39:13.22  >       MONITOR
ABPS0836I 110 03:39:13.22  >        SYNTAX VALUE="USMLST" SUBSTITUTE="USMLST2"/
ABPS0836I 110 03:39:13.22  >       /MONITOR
ABPS0836I 110 03:39:13.22  >     /UTILITY
ABPS0836I 110 03:39:13.22  >   /PRACTICE
ABPS0836I 110 03:39:13.22  >
```

DISPLAY SESSIONS

Use this command to list information on currently active sessions.

Issue the **MODIFY** command from SDSF as follows: `/F started_task_name,DISPLAY SESSIONS`

This command produces the following report in the JOBLOG on currently active sessions:

```
ABPS0700I 069 10:42:50.10 TCB: 008CAB00  SESSION REPORT *00000001*
ABPS0701I 069 10:42:50.10      *00000001* SESS: 25C40048-00000002-I-PDRICKB -T0897680-00E0-PDRICKB
ABPS0702I 069 10:42:50.10      *00000001*       STATUS: SIGNED ON
ABPS0703I 069 10:42:50.10      *00000001*       STARTED: 03-10-2009  14:42:42  UTC
ABPS0701I 069 10:42:50.10      *00000001* SESS: 25C40208-00000001-I-PDRICKA -T0897697-00EB-PDRICKA
ABPS0702I 069 10:42:50.10      *00000001*       STATUS: SIGNED ON
ABPS0703I 069 10:42:50.10      *00000001*       STARTED: 03-10-2009  14:42:37  UTC

ABPS0700I 069 10:43:13.68 TCB: 008CAB00  SESSION REPORT *00000002*
ABPS0704I 069 10:43:13.68      *00000002* No active sessions found
```

DISPLAY TRACE

Use this command to capture trace information for the specified started task. This information is written to a SNAPTRC data set that is allocated to the started task. Trace information is primarily used for diagnosing problems. Only issue this command when directed to do so by Support.

DUMP

Use this command to perform an SVC dump of the started task address space. Usually, a dump is produced at the request of Support to collect error information for analysis. You can find the location of the dump data set in the system log. If the started task is unresponsive, you can produce a dump of other DB2 UET address spaces. For information about producing dumps of multiple address spaces, see "Producing dumps for diagnostic use" on page 439.

HELP Use this command to list all of the z/OS console commands that are supported for the started task in the SYSPRINT data set for the started task. The list indicates the correct syntax for these commands.

LIST PRACTICE[*,ssid*]

Use this command to generate a practice report that lists all of the practices that are defined in the policy for the specified subsystem ID. The current active practice is marked.

Issue the **MODIFY** command from SDSF as follows: `/F started_task_name,LIST PRACTICE,LA1A`

This command produces the following report in SYSPRINT for DB2 subsystem LA1A:

```
ABPS0814I 110 02:17:49.96 Command issued: LIST PRACTICE,LA1A

ABPS0836I 110 02:17:49.97 DB2 UTILITIES ENHANCEMENT TOOL practice report
ABPS0836I 110 02:17:49.97 ---------------------------------------------
ABPS0836I 110 02:17:49.97  DB2 SSID: LA1A  ACTIVE PRACTICE: PRACTICE_1
ABPS0836I 110 02:17:49.97  DB2 SSID: LA1A  ACTIVE PRACTICE: PRACTICE_2
ABPS0836I 110 02:17:49.97  DEFINED PRACTICE: PRACTICE_1
ABPS0836I 110 02:17:49.97  DEFINED PRACTICE: PRACTICE_2
```

--REFRESH POLICY

Warning! Use of this command is not recommended. The highly recommended procedure for refreshing the policy is to stop and restart the started task with the new policy rules in place. When you use this command, the DB2 UET started task has no control over when the policy information is refreshed. Therefore, when this command is submitted, the address space that contains the policy rules is immediately refreshed, and results are unpredictable. That is, if DB2 UET is monitoring active DB2 utilities when you issue this command, the DB2 utilities were started with one set of rules, but will complete under a different set of rules.

Use this command to refresh the address space that contains the DB2 UET policy rules. While this online command is convenient, use it only in rare instances when the started task cannot be taken down. Perform this type of policy refresh only when there is absolutely no DB2 utility activity on the subsystems that DB2 UET is monitoring.

Issue the **MODIFY** command from SDSF as follows:
`/F started_task_name,--REFRESH POLICY`
This command produces the following report in SYSPRINT for the started task:

```
ABPS0814I 036 16:10:36.06 Command issued: --REFRESH POLICY
ABPS0604W 036 16:10:36.13 DSNUTILB interception for DB2 SSID=RA1B enabled, DB2 subsystem is not active.
ABPS0830I 036 16:10:36.13 DSNUTILB Intercept Policy:
ABPS0831I 036 16:10:36.13  DB2 SSID: RA1B  ACTION: BLOCK_AND_CANCEL_THREADS
```

STOP [FORCE]

Use this command to stop the specified started task. The `/F started_task_name,STOP` operator command is equivalent to the standard `/P started_task_name` operator command. If you want to stop the started task immediately, before it completes its current processing, you can add the optional **FORCE** option after the **STOP** command. To separate **FORCE** from **STOP**, use only a single space (not a comma).

TERMINATE SESSION,SESS=*session_address*

If a DB2 UET batch job, ISPF interface session, intercepted DSNUTILB utility execution, or ABPMAINT job terminates abnormally without ending its session with the DB2 UET started task, you can use this command to force the termination of the session. For the SESS value in this command, specify a valid session address that is an 8-digit hexadecimal number. (A hexadecimal number can contain only the characters 0 through 9 and A through F.) You should be able to find this session address in an ABPS0101I message. After you issue the command, look for message ABPS0103I to determine whether the session terminated. You might want to use this command, for example, when message ABPS5113I is issued. This message indicates that a DB2 utility cannot be restarted because its worklist is in use by another utility. If the other utility has terminated abnormally but is still associated with an active "owning session," you can terminate the owning session by using this command. You should then be able to perform the restart operation.

Related tasks:

"Stopping the started task" on page 306
Occasionally, you might to need to stop the DB2 UET started task to perform a task such as applying product maintenance.

User exit details

These topics provide reference information for each type of DB2 UET user exit that you can optionally define: a security exit, a pre-cancel exit, and a post-cancel exit. This information is primarily intended for the assembler-language programmers who will develop the exits.

For each exit, a description of when the exit is called and the DSECT are provided. Review this information prior to creating a user exit for your environment. You should not change any of the DSECTs.

Security exit

You can create an optional security user exit to control user access to DB2 UET ISPF panels and thread-management functions. The exit will apply to a single DB2 UET configuration. If you do not create a security exit, all users will have access to all panels and thread-management functions.

A security user exit must be written in assembler language. To write a security exit, you should be an experienced assembler-language programmer. You should also have experience with system-level user exits and DB2 and a comprehensive knowledge of security issues. If the exit is not carefully designed and written, it might degrade the performance of your DB2 UET system.

A sample user exit (ABPXSE00) and corresponding DSECT (ABPAPISE) are provided in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization. You can use

the sample exit as a basis for implementing your own security exit. Be sure to preserve the original sample exit and DSECT in case you need to refer back to them later or reuse them.

The sample exit implements a simple security scheme that limits the ability of users to perform normal and escalated thread cancelations and the three thread-blocker actions (BLOCK_THREADS, ALLOW_THREADS, and DELETE_BLOCKERID). It also can limit user access to certain DB2 UET ISPF panels based on user IDs. Depending on your site's security requirements, the exit that you develop might be very different from and much more complex than the sample exit. For example, you could create an exit that calls the z/OS Resource Access Control Facility (RACF) to determine a user's authority to cancel threads.

Whenever you upgrade the product to a new version or release or apply a PTF, be sure to check the SABPSAMP library for the latest sample exit and DSECT. If the DSECT changes, you will need to reassemble and relink your user exit.

Related concepts:

"Authorization requirements for the started task" on page 31
Make sure that the DB2 UET started task runs under a user ID that has the required authority. To control security at the user level, you must create a security exit.

Related tasks:

"Creating a security exit (optional)" on page 106
If you want to control user access to product commands for intercept processing and to certain ISPF panels, you can create a security user exit. This customization step is optional.

When is the security exit called?

The DB2 UET started task calls the security exit at started task initialization and termination and when users perform certain actions.

Functions for which the sample security exit is called:

INIT The user exit is called once when the started task initializes.

SIGNON
The user exit is called when a user starts a DB2 UET session.

SIGNOFF
The user exit is called when a user ends a DB2 UET session.

CHECK
The user exit is called whenever a user attempts any of the following actions:

Action	Resource Class	Resource Name
Perform a normal thread cancelation, which uses the DB2 -CANCEL THREAD command, from any DB2 UET interface	COMMAND	CANCEL
Perform an escalated cancelation that uses the z/OS Cancel command	COMMAND	ECANCEL
Block new threads from forming on the DB2 objects for which threads are being canceled until after your batch utility or utilities run	COMMAND	BLOCK_THREADS

Action	Resource Class	Resource Name
End thread blocking and allow new threads to form on DB2 objects once again	COMMAND	ALLOW_THREADS.*userid* where *userid* is the user ID of the user who originally performed the BLOCK_THREADS action
Delete all data that is associated with a specific *blocker_id* (a user-defined ID for a thread-blocker function) from the DB2 table that stores thread-blocker data	COMMAND	DELETE_BLOCKERID.*userid* where *userid* is the user ID of the user who originally performed the BLOCK_THREADS action
Display and use the Specify Thread Filter Criteria panel in the ISPF interface	PANEL	ABPLTHD1
Display and use the Thread Summary Report panel in the ISPF interface	PANEL	ABPLTHD2
Display and use the Administrator Functions menu panel in the ISPF interface	PANEL	ABPADM0
Display the Display System Status panel in the ISPF interface	PANEL	ABPADM1
Display and use the Control System panel in the ISPF interface	PANEL	ABPADM2

TERM The user exit is called once when the started task terminates.

Note: If DB2 UET cannot load the exit or if errors occur when the exit runs, the started task will terminate.

DSECT ABPAPISE

Learn about the DSECT ABPAPISE that is provided for the security exit. This DSECT should not be changed.

Register 1 (R1) contains the address of the control block for the application programming interface (API) that enables communication between the DB2 UET started task and the user exit code. The sample exit ABPXSE00 maps this address to the DSECT ABPAPISE. The DSECT describes the layout of the API control block in memory. The values for the fields in the DSECT are set by the started task before the user exit is called.

Important: Do not change the DSECT. Maintain the DSECT in its original state to ensure that it properly defines the layout of the memory for the API control block.

The DSECT is available in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization. It is divided into an Environmental section and Exit-Specific Data section, as follows:

```
*------------------------------------------------------------------------*
*                                                                        *
*     PRODUCT: DB2 UTILITIES ENHANCEMENT TOOL                            *
* MODULE NAME: ABPAPISE                                                  *
* DESCRIPTION: USER EXIT (SECURITY) API                                  *
*                                                                        *
*    POINTED TO BY:  (R1) WHEN THE SECURITY EXIT IS CALLED               *
```

```
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
*  5655-T58                                                           *
*  (C) COPYRIGHT ROCKET SOFTWARE, INC. 2002, 2016 ALL RIGHTS          *
*  RESERVED.                                                          *
*                                                                     *
*---------------------------------------------------------------------*
          TITLE 'ABPAPISE - USER EXIT (SECURITY) API'
 ABPAPISE DSECT
 *------- ENVIRONMENTAL ------------------------------------------------*
 UXSEFC   DS    X                       FUNCTION CODE:
 UXSE_FC_INIT    EQU X'80' 1... ....     - INIT CALL
 UXSE_FC_TERM    EQU X'40' .1.. ....     - TERM CALL
 UXSE_FC_CHECK   EQU X'20' ..1. ....     - CHECK CALL
 UXSE_FC_SIGNON  EQU X'10' ...1 ....     - SIGNON CALL
 UXSE_FC_SIGNOFF EQU X'08' .... 1...     - SIGNOFF CALL
 *               EQU X'04' .... .1..       ** UNUSED **
 *               EQU X'02' .... ..1.       ** UNUSED **
 *               EQU X'01' .... ...1       ** UNUSED **

 UXSEUFW  DS    F                       USER FULLWORD (SET IN INIT CALL)

 UXSERC   DS    F                       RETURN CODE
 UXSE_RC_PASS   EQU 0                    - PASS
 UXSE_RC_FAIL   EQU 8                    - FAIL
 UXSE_RC_ERROR  EQU 12                   - SEVERE ERROR

 UXSEUSER DS    CL8                     USERID

 UXSESEB  DS    F                       SECURITY ENVIRONMENT BLOCK

 UXSEENV  DS    C                       ENVIRONMENT:
 UXSE_ENV_BAT EQU   C'B'                 - BATCH
 UXSE_ENV_ISP EQU   C'I'                 - ISPF
 UXSE_ENV_MNT EQU   C'M'                 - MAINTENANCE UTILITY
 UXSE_ENV_UTL EQU   C'U'                 - DSNUTILB INTERCEPT

 *------- EXIT-SPECIFIC DATA -------------------------------------------*
 UXSERSCL DS    CL8                     RESOURCE CLASS
 UXSERSNM DS    CL44                    RESOURCE NAME

 UXSEACLV DS    X                       ACCESS LEVEL:
 UXSE_ACLV_READ    EQU X'01' .... ...1   - READ
 UXSE_ACLV_UPDATE  EQU X'02' .... ..1.   - UPDATE
 UXSE_ACLV_CONTROL EQU X'04' .... .1..   - CONTROL
 UXSE_ACLV_ALL     EQU X'07' .... .111   - ALL

 ABPAPISE$ EQU  *-ABPAPISE               LENGTH
 ******************************** Bottom of Data ********************************
```

Environmental section

This section defines the layout of the control block for the following security exit fields:

UXSEFC
(FUNCTION CODE). Specifies the logical function for which the exit is being called. This function can be one of the following:

- INIT: User exit initialization
- TERM: User exit termination
- CHECK: A user action for which security is checked, such as an attempt to cancel a thread or view a certain ISPF panel
- SIGNON: A user's initiation of a DB2 UET session

- SIGNOFF: A user's ending of a DB2 UET session

UXSEUFW
(USER FULLWORD). Specifies the address of persistent storage that the exit can optionally acquire when the started task initializes (at the INIT function call). This storage can then be used for subsequent CHECK, SIGNON, and SIGNOFF function calls. The storage is freed when the started task terminates (at the TERM function call) or when the system is shut down.

UXSERC
(RETURN CODE). Specifies the return code that the exit returns to DB2 UET to indicate whether the action that is specified by the INIT, TERM, CHECK, SIGNON, or SIGNOFF function call passed or failed security checking. The return code can be one of the following:

- RC=0 indicates that the action that is specified by the function call is allowed.
- RC=8 indicates that the action that is specified by the function call is *not* allowed.
- RC=12 indicates the occurrence of a severe error that always causes the started task to terminate.

UXSEUSER
(USERID) Specifies the user ID of the user whose authority is being checked. The exit provides this information for the SIGNON, SIGNOFF, or CHECK function calls.

UXSESEB
(SECURITY ENVIRONMENT BLOCK). Specifies the address of a unique block of memory such as a user-specific security control block. The exit obtains and saves this address during the SIGNON function call. The exit can then provide this address for each CHECK and SIGNOFF function call. For example, if you are using RACF security, this value might be the address a users Accessor Environment Element (ACEE).

UXSEENV
(ENVIRONMENT). Specifies the DB2 UET interface that the user is using. The exit provides this information for SIGNON, SIGNOFF, and CHECK function calls. The interface value can be one of the following:

- B for the batch interface
- I for the ISPF interface
- M for the ABPMAINT maintenance utility
- U for the DSNUTILB intercept

Exit-specific data section

This section describes the following exit-specific data fields. These fields indicate the resource to which the security exit is controlling access. This information is provided for CHECK function calls only.

UXSERSCL
(RESOURCE CLASS). Specifies one of the following classes, depending on the type of resource for which security is being checked:

- COMMAND for a cancel command or thread-blocker action
- PANEL for a DB2 UET ISPF pane

UXSERSNM

(RESOURCE NAME). Specifies the name of a DB2 UET resource for which security is being checked:

- If the resource class is COMMAND, the resource name will be one of the following:
  - CANCEL for the DB2 -CANCEL THREAD command
  - ECANCEL for an escalated cancel command
  - BLOCK_THREADS for blocking new threads from forming on DB2 objects as part of a batch thread-cancelation job step
  - ALLOW_THREADS.*userid* for ending thread blocking and allowing new threads to form on DB2 objects once again (where *userid* is the user ID of the user who is attempting the ALLOW_THREADS action)
  - DELETE_BLOCKERID.*userid* for removing all data that is associated with a specific *blocker_id* from the DB2 table that stores thread-blocker data (where *userid* is the user ID of the user who is attempting the DELETE_BLOCKERID action)
- If the resource class is PANEL, the resource name will be one of the following panel IDs:
  - ABPLTHD1 for the Thread Filter Criteria panel
  - ABPLTHD2 for the Thread Summary Report panel
  - ABPADM0 for the Administrator Functions menu
  - ABPADM1 for the Display System Status panel
  - ABPADM2 for the Control System panel

UXSEACLV

(ACCESS LEVEL). Specifies the users level of access to the specified resource:

- READ allows the user to view the specified resource but not to update it.
- UPDATE allows the user to view and update the resource (for example, change information on a panel, issue a Cancel command, or block new threads from forming).
- CONTROL allows the user to administer the resource.
- ALL allows the user to view, update, and administer the resource.

Pre-cancel exit

You can create an optional pre-cancel user exit to perform any processing that you consider to be necessary prior to canceling DB2 threads from theDB2 UET ISPF or batch interface. The exit will apply to a single DB2 UET configuration.

A pre-cancel exit must be written in assembler language. To create a pre-cancel exit, you should be an experienced assembler-language programmer. You should also have experience with system-level user exits and DB2. If the exit is not carefully designed and written, it might degrade the performance of your DB2 UET system.

A sample user exit (ABPXPR00) and corresponding DSECT (ABPAPIPR) are provided in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization. You can use the sample exit as a basis for creating your own pre-cancel exit. Be sure to preserve the original sample exit and DSECT in case you need to refer back to them later or reuse them.

The sample exit is a general template for implementing pre-cancel processing. The exit is called for each thread that is canceled with the normal cancel command (DB2 -CANCEL THREAD) or with the escalated cancel command. You will need to add code to specify the type of processing that you want to occur.

Whenever you upgrade the product to a new version or release or apply a PTF, check the SABPSAMP library for the latest exit and DSECT. If the DSECT changes, you will need to reassemble and relink your user exit.

Related tasks:

"Creating a pre- or post-cancel exit (optional)" on page 107
If you need to perform additional processing prior to or after thread cancelations, you can create a pre-cancel exit, post-cancel exit, or both. This customization step is optional.

When is the pre-cancel exit called?

The DB2 UET started task calls the pre-cancel exit after the security exit, if defined, completes and only for the described functions.

Functions for which the pre-cancel exit is called:

INIT The exit is called once when the started task initializes.

CHECK
The exit is called after the security exit completes its processing for a cancel command and *before* the thread is canceled.

TERM The exit is called once when the started task terminates.

Note: If DB2 UET cannot load the exit or if errors occur when the exit runs, the started task will terminate.

DSECT ABPAPIPR

Learn about the DSECT ABPAPIPR that is provided for the pre-cancel exit. This DSECT should not be changed.

Register 1 (R1) contains the address of the control block for the application programming interface (API) that enables communication between the DB2 UET started task and the user exit code. The sample exit ABPXPR00 maps this address to the DSECT ABPAPIPR. The DSECT describes the layout of the API control block in memory. The values for the fields in the DSECT are set by the started task before the user exit is called.

Important: Do not change the DSECT. Maintain the DSECT in its original state to ensure that it properly defines the layout of the memory for the API control block.

The DSECT is available in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization. It is divided into an Environmental section and Exit-specific data section, as follows:

```
*------------------------------------------------------------------*
*                                                                  *
*     PRODUCT: DB2 UTILITIES ENHANCEMENT TOOL                      *
* MODULE NAME: ABPAPIPR                                            *
* DESCRIPTION: USER EXIT (PRE-CANCEL) API                          *
*                                                                  *
*    POINTED TO BY:  (R1) WHEN THE PRE-CANCEL EXIT IS CALLED       *
*                                                                  *
*------------------------------------------------------------------*
```

```
*                                                                     *
*  5655-T58                                                           *
*  (C) COPYRIGHT ROCKET SOFTWARE, INC. 2002, 2016 ALL RIGHTS          *
*  RESERVED.                                                          *
*                                                                     *
*---------------------------------------------------------------------*
          TITLE 'ABPAPIPR - USER EXIT (PRE-CANCEL) API'
 ABPAPIPR DSECT

 *------- ENVIRONMENTAL ------------------------------------------------*
 UXPRFC   DS    X                      FUNCTION CODE:
 UXPR_FC_INIT    EQU X'80' 1... ....     - INIT CALL
 UXPR_FC_TERM    EQU X'40' .1.. ....     - TERM CALL
 UXPR_FC_CHECK   EQU X'20' ..1. ....     - CHECK CALL
 *               EQU X'10' ...1 ....       ** UNUSED **
 *               EQU X'08' .... 1...       ** UNUSED **
 *               EQU X'04' .... .1..       ** UNUSED **
 *               EQU X'02' .... ..1.       ** UNUSED **
 *               EQU X'01' .... ...1       ** UNUSED **

 UXPRUFW  DS    F                      USER FULLWORD (SET IN INIT CALL)

 UXPRRC   DS    F                      RETURN CODE
 UXPR_RC_PASS   EQU 0                   - PASS
 UXPR_RC_FAIL   EQU 8                   - FAIL
 UXPR_RC_ERROR  EQU 12                  - SEVERE ERROR

 UXPRUSER DS    CL8                    USERID

 UXPRSEB  DS    F                      SECURITY ENVIRONMENT BLOCK

 UXPRENV  DS    C                      ENVIRONMENT:
 UXPR_ENV_BAT EQU   C'B'                - BATCH
 UXPR_ENV_ISP EQU   C'I'                - ISPF
 UXPR_ENV_MNT EQU   C'M'                - ABPMAINT UTILITY
 UXPR_ENV_UTL EQU   C'U'                - DSNUTILB INTERCEPT

*------- EXIT-SPECIFIC DATA -------------------------------------------*
 * UXPR_ACE_TOKEN      DS F     ACE                 DEPRECATED  PRD0210
 UXPR_CONNECT_TYPE   DS F     CONNECTION TYPE
 * UXPR_FTDD_TOKEN     DS F     THREAD DETAIL TOKEN  DEPRECATED PRD0210
 UXPR_THREAD_TOKEN   DS A     THREAD TOKEN
 UXPR_ASID_BINARY    DS H     BINARY ASID
 UXPR_SQL_REQ_COUNT  DS H     SQL REQUEST COUNT
 * UXPR_STATUS_FLAGS   DS X     STATUS FLAG BYTE     DEPRECATED PRD0210
 UXPR_ACTIVE_IN_DB2  DS C     ACTIVE IN DB2
 UXPR_STATUS         DS CL2   THREAD STATUS
 UXPR_ASID           DS CL4   ASID
 UXPR_IP_PORT        DS CL5   IP PORT
 UXPR_JOBID          DS CL8   JOBID
 UXPR_AUTH_ID        DS CL8   AUTH ID
 UXPR_PLAN           DS CL8   PLAN
 UXPR_MEMBER_NAME    DS CL8   MEMBER NAME
 UXPR_ORIG_AUTH_ID   DS CL8   ORIGINAL AUTH ID
 UXPR_CONN_ID        DS CL8   CONNECTION TYPE CHAR
 UXPR_PROGRAM_NAME   DS CL128 PROGRAM NAME                      ABP0434
 UXPR_JOBNAME        DS CL8   JOBNAME
 UXPR_NAME           DS CL12  CONNECTION NAME
 UXPR_CORR_ID        DS CL12  CORRELATION ID
 UXPO_IP_ADDRESS     DS CL128 IP ADDRESS                        ABP0121
 UXPO_LOCATION_NAME  DS CL128 LOCATION NAME                     ABP0121
 UXPR_COLLECTION_ID  DS CL128 COLLECTION NAME
 UXPR_PACKAGE_NAME   DS CL128 PACKAGE NAME

 ABPAPIPR$ EQU  *-ABPAPIPR              LENGTH
******************************** Bottom of Data ********************************
```

Environmental section

This section defines the layout of the control block for the following pre-cancel exit fields:

UXPRFC
(FUNCTION CODE). Specifies the logical function for which the exit is being called. This function can be one of the following:

- INIT: User exit initialization
- TERM: User exit termination
- CHECK: Prior to canceling a thread

UXPRUFW
(USER FULLWORD). Specifies the address of persistent storage that the exit can optionally acquire when the started task initializes (at the INIT function call). This storage can then be used for subsequent CHECK function calls. The storage is freed when the started task terminates (at the TERM function call) or when the system is shut down.

UXPRRC
(RETURN CODE). Specifies the return code that the exit returns to DB2 UET to indicate whether the action that is specified by the INIT, TERM, or CHECK function call passed or failed security checking. The return code can be one of the following:

- RC=0 indicates that the action that is specified by the function call is allowed.
- RC=8 indicates that the action that is specified by the function call is *not* allowed. For batch thread-cancelation jobs only, this return code does not affect the return code that is issued by the batch interface program (ABPBMAIN) for a job step. This behavior allows the pre-cancel exit to perform a CHECK function that returns RC=8; the return code can then be used to control certain actions (for example, preventing certain threads from being canceled).
- RC=12 indicates the occurrence of a severe error that always causes the started task to terminate.

UXPRUSER
(USERID). Specifies the user ID of the user who issued the cancel command. The exit provides this information only for CHECK function calls.

UXPRSEB
(SECURITY ENVIRONMENT BLOCK). Specifies the address of a unique block of memory such as a user-specific security control block. The security exit obtains this address. The address is passed to the pre-cancel exit during CHECK function calls only.

UXPRENV
(ENVIRONMENT). Specifies the DB2 UET interface from which the cancel command was issued. The exit provides this information only for CHECK function calls. The interface value can be one of the following:

- B for the batch interface
- I for the ISPF interface
- M for the ABPMAINT utility (not applicable to cancel processing)
- U for the DSNUTILB intercept

Exit-specific data section

This section defines the following fields that provide details about a thread that was selected for cancelation and for which the pre-cancel exit was called. This information is provided for CHECK function calls only.

UXPR_CONNECT_TYPE
(CONNECTION TYPE). Specifies the numeric identifier for the connection type. A connection type can be a system region or program from which the thread originated or the type of distributed access protocols that the thread is using. The name of the connection type in character format is provided in the CONNECTION TYPE CHAR field.

UXPR_THREAD_TOKEN
(THREAD TOKEN). Specifies the thread token value for the thread. A thread token value is a unique numeric identifier that DB2 assigns to each active thread on a DB2 subsystem.

UXPR_ASID_BINARY
(BINARY ASID). Specifies the address space identifier (in binary format) for the address space from which the thread originated.

UXPR_SQL_REQ_COUNT
(SQL REQUEST COUNT). Specifies the number of SQL calls that the thread performed.

UXPR_ACTIVE_IN_DB2
(ACTIVE IN DB2). Specifies the thread status:

- In-DB2: The thread is active and running in DB2.
- In-APPL: The thread is not active in DB2.
- DEF-TERM: The thread is active and in deferred termination processing.
- Q-DEFTERM: The thread is active and is queued for deferred termination processing.

UXPR_STATUS
(THREAD STATUS). Specifies the DB2 connection status code for the thread. Many codes are available and are described in the message DSNV404I in the *DB2 Version 9.1 for z/OS Messages* manual.

UXPR_ASID
(ASID). Specifies the address space identifier (in character hex format) for the address space from which the thread originated.

UXPR_IP_PORT
(IP PORT). Specifies the IP port number of the Distributed Database Facility (DDF) requestor from which the threads originated.

UXPR_JOBID
(JOBID). Specifies the unique identifier that the z/OS system generated for the job from which the thread originated.

UXPR_AUTH_ID
(AUTH ID). Specifies the current authorization ID of the process that is associated with the thread. An authorization ID is a character string that can be verified for connection to DB2. If the SET CURRENT SQLID statement was used to change the original authorization ID of the process, this field displays the ID that is currently used and the **ORIGINAL AUTH ID** field displays the authorization ID that was initially used.

UXPR_PLAN
(PLAN). Specifies the name of the DB2 plan under which the thread is running.

UXPR_MEMBER_NAME
(MEMBER NAME). Specifies the data-sharing member name of the subsystem that the thread is referencing within a data-sharing group.

UXPR_ORIG_AUTH_ID
(ORIGINAL AUTH ID). Specifies the original authorization identifier of the process for which the threads were initially established.

UXPR_CONN_ID
(CONNECTION TYPE CHAR). Specifies the connection type in character format. A connection type can be a system region or program from which the thread originated or the type of distributed access protocols that the thread is using.

- CAF: The thread originates from a Call Attach Facility (CAF) user.
- CICS: The thread originates from a CICS region.
- DBAT_DRD: The thread is a distributed database access thread that uses distributed relational database access protocol (DRDA).
- DBAT_PP: The thread is a distributed database access thread that uses distributed database private protocols (DDF).
- IMSBMP: The thread originates from an IMS Batch Message Processing (BMP) program.
- IMSCTL: The thread originates from an IMS control region.
- IMSDLIBT: The thread originates from an IMS Data Language/I (DL/I) batch region.
- IMSMPP: The thread originates from an IMS message region.
- IMSTBMP: The thread originates from an IMS transaction BMP.
- RRSAF: The thread originates from the Recoverable Resource Manager Service attachment facility.
- TSO: The thread originates from a TSO/E interactive or batch user.
- UTILITY: The thread originates from a DB2 utility.

UXPR_PROGRAM_NAME
(PROGRAM NAME). Specifies the name of the program that is currently running with the thread.

UXPR_JOBNAME
(JOBNAME). Specifies the name of the job from which the thread originated. This name is specified in the job card.

UXPR_NAME
(CONNECTION NAME). Specifies the name of the connection. This name is the same as the **Name** value that is returned from the DB2 -DISPLAY THREAD command.

UXPR_CORR_ID
(CORRELATION ID). Specifies the correlation ID of the thread. A correlation ID is an identifier that is associated with a specific thread. For example, it can be a TSO user ID or the job name.

UXPO_IP_ADDRESS
(IP ADDRESS). Specifies the IP address of the DDF requestor from which the thread originated. An IP address is a unique address of a location on the Internet.

UXPO_LOCATION_NAME
(LOCATION NAME). Specifies the name of the remote DDF requestor location from which the thread originated.

UXPR_COLLECTION_ID
(COLLECTION NAME). Specifies the collection identifier for the group of packages that includes the package under which the thread was running.

UXPR_PACKAGE_NAME
(PACKAGE NAME). Specifies the name of the DB2 package under which the thread is running.

Post-cancel exit

You can create an optional post-cancel user exit to perform any processing that you consider to be necessary after canceling DB2 threads from the DB2 UET ISPF or batch interface. This exit applies to a single DB2 UET configuration.

A post-cancel exit must be written in assembler language. To create a post-cancel exit, you should be an experienced assembler language programmer. You should also have experience with system-level user exits and DB2. If the exit is not carefully designed and written, it might degrade the performance of your DB2 UET system.

A sample user exit (ABPXPO00) and corresponding DSECT (ABPAPIPO) are provided in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization. You can use the sample exit as a basis for creating your own post-cancel exit. Be sure to preserve the original sample exit and DSECT in case you need to refer back to them later or reuse them.

The sample exit is a general template for implementing post-cancel processing. It is called for each thread that a user cancels with the normal cancel command (DB2 -CANCEL THREAD) or with the escalated cancel command (z/OS Cancel command). You will need to add code to specify the type of processing that you want to occur.

Whenever you upgrade the product to a new version or release or apply a PTF, check the SABPSAMP library for the latest exit and DSECT. If the DSECT changes, you will need to reassemble and relink your user exit.

Related tasks:

"Creating a pre- or post-cancel exit (optional)" on page 107
If you need to perform additional processing prior to or after thread cancelations, you can create a pre-cancel exit, post-cancel exit, or both. This customization step is optional.

When is the post-cancel exit called?

The DB2 UET started task calls the post-cancel exit for a thread after the security and pre-cancel exits, if defined, complete and only for the described functions.

Functions for which the post-cancel exit is called:

INIT The exit is called once when the started task initializes.

CHECK
The exit is called immediately *after* a thread is canceled.

TERM The exit is called once when the started task terminates.

Note: If DB2 UET cannot load the exit or if errors occur when the exit runs, the started task will terminate.

DSECT ABPAPIPO

Learn about the DSECT ABPAPIPO that is provided for the post-cancel exit. This DSECT should not be changed.

Register 1 (R1) contains the address of the control block for the application programming interface (API) that enables communication between the DB2 UET started task and the user exit code. The sample exit ABPXPO00 maps this address to the DSECT ABPAPIPO. The DSECT describes the layout of the API control block in memory. The values for the fields in the DSECT are set by the started task before the user exit is called.

Important: Do not change the DSECT. Maintain the DSECT in its original state to ensure that it properly defines the layout of the memory for the API control block.

The DSECT is available in the *hlq.mlq*.SABPSAMP library, where *hlq* is the high-level qualifier and *mlq* is the mid-level qualifier that you specified during customization. It is divided into an Environmental section and Exit-specific data section, as follows:

```
*------------------------------------------------------------------------*
*                                                                        *
*     PRODUCT: DB2 UTILITIES ENHANCEMENT TOOL                            *
* MODULE NAME: ABPAPIPO                                                  *
* DESCRIPTION: USER EXIT (POST-CANCEL) API                               *
*                                                                        *
*    POINTED TO BY:  (R1) WHEN THE POST-CANCEL EXIT IS CALLED            *
*                                                                        *
*------------------------------------------------------------------------*
*                                                                        *
*  5655-T58                                                              *
*  (C) COPYRIGHT ROCKET SOFTWARE, INC. 2002, 2016 ALL RIGHTS             *
*  RESERVED.                                                             *
*                                                                        *
*------------------------------------------------------------------------*
          TITLE 'ABPAPIPO - USER EXIT (POST-CANCEL) API'
 ABPAPIPO DSECT

 *------- ENVIRONMENTAL -------------------------------------------------*
 UXPOFC   DS    X                      FUNCTION CODE:
 UXPO_FC_INIT    EQU X'80' 1... ....     - INIT CALL
 UXPO_FC_TERM    EQU X'40' .1.. ....     - TERM CALL
 UXPO_FC_CHECK   EQU X'20' ..1. ....     - CHECK CALL
 *               EQU X'10' ...1 ....       ** UNUSED **
 *               EQU X'08' .... 1...       ** UNUSED **
 *               EQU X'04' .... .1..       ** UNUSED **
 *               EQU X'02' .... ..1.       ** UNUSED **
 *               EQU X'01' .... ...1       ** UNUSED **

 UXPOUFW  DS    F                      USER FULLWORD (SET IN INIT CALL)

 UXPORC   DS    F                      RETURN CODE
 UXPO_RC_PASS   EQU 0                   - PASS
 UXPO_RC_FAIL   EQU 8                   - FAIL
 UXPO_RC_ERROR  EQU 12                  - SEVERE ERROR

 UXPOUSER DS    CL8                    USERID

 UXPOSEB  DS    F                      SECURITY ENVIRONMENT BLOCK

 UXPOENV  DS    C                      ENVIRONMENT:
 UXPO_ENV_BAT EQU   C'B'                - BATCH
```

```
 UXPO_ENV_ISP EQU   C'I'                  - ISPF
 UXPO_ENV_MNT EQU   C'M'                  - ABPMAINT UTILITY
 UXPO_ENV_UTL EQU   C'U'                  - DSNUTILB INTERCEPT

*------- EXIT-SPECIFIC DATA ------------------------------------------*
 * UXPO_ACE_TOKEN      DS F     ACE                 DEPRECATED  PRD0210
 UXPO_CONNECT_TYPE   DS F     CONNECTION TYPE
 * UXPO_FTDD_TOKEN     DS F     THREAD DETAIL TOKEN DEPRECATED  PRD0210
 UXPO_THREAD_TOKEN   DS A     THREAD TOKEN
 UXPO_ASID_BINARY    DS H     BINARY ASID
 UXPO_SQL_REQ_COUNT  DS H     SQL REQUEST COUNT
 * UXPO_STATUS_FLAGS   DS X     STATUS FLAG BYTE    DEPRECATED  PRD0210
 UXPO_ACTIVE_IN_DB2  DS C     ACTIVE IN DB2
 UXPO_STATUS         DS CL2   THREAD STATUS
 UXPO_ASID           DS CL4   ASID
 UXPO_IP_PORT        DS CL5   IP PORT
 UXPO_JOBID          DS CL8   JOBID
 UXPO_AUTH_ID        DS CL8   AUTH ID
 UXPO_PLAN           DS CL8   PLAN
 UXPO_MEMBER_NAME    DS CL8   MEMBER NAME
 UXPO_ORIG_AUTH_ID   DS CL8   ORIGINAL AUTH ID
 UXPO_CONN_ID        DS CL8   CONNECTION TYPE CHAR
 UXPO_PROGRAM_NAME   DS CL128 PROGRAM NAME                    ABP0434
 UXPO_JOBNAME        DS CL8   JOBNAME
 UXPO_NAME           DS CL12  CONNECTION NAME
 UXPO_CORR_ID        DS CL12  CORRELATION ID
 UXPO_IP_ADDRESS     DS CL128 IP ADDRESS                      ABP0121
 UXPO_LOCATION_NAME  DS CL128 LOCATION NAME                   ABP0121
 UXPO_COLLECTION_ID  DS CL128 COLLECTION NAME
 UXPO_PACKAGE_NAME   DS CL128 PACKAGE NAME

 ABPAPIPO$ EQU  *-ABPAPIPO              LENGTH
 ******************************** Bottom of Data ********************************
```

Environmental section

This section defines the layout of the control block for the following post-cancel exit fields:

UXPOFC
(FUNCTION CODE). Specifies the logical function for which the exit is being called. This function can be one of the following:

- INIT: User exit initialization
- TERM: User exit termination
- CHECK: After canceling a thread

UXPOUFW
(USER FULLWORD). Specifies the address of persistent storage that the exit can optionally acquire when the started task initializes (at the INIT function call). This storage can then be used for subsequent CHECK function calls. The storage is freed when the started task terminates (at the TERM function call) or when the system is shut down.

UXPORC
(RETURN CODE). Specifies the return code that the exit returns to DB2 UET to indicate whether the action that is specified by the INIT, TERM, or CHECK function call passed or failed security checking. The return code can be one of the following:

- RC=0 indicates that the action that is specified by the function call is allowed.
- RC=8 indicates that the action that is specified by the function call is *not* allowed.

- RC=12 indicates the occurrence of a severe error that always causes the started task to terminate.

UXPOUSER
(USERID). Specifies the user ID of the user who issued the cancel command. The exit provides this information only for CHECK function calls.

UXPOSEB
(SECURITY ENVIRONMENT BLOCK). Specifies the address of a unique block of memory such as a user-specific security control block. The security exit obtains this address. The address is passed to the post-cancel exit during CHECK function calls only.

UXPOENV
(ENVIRONMENT). Specifies the DB2 UET interface from which the cancel command was issued. The exit provides this information only for CHECK function calls. The interface value can be one of the following:

- B for the batch interface
- I for the ISPF interface
- M for the ABPMAINT utility (not applicable to cancel processing)
- U for the DSNUTILB intercept

Exit-specific data section

This section defines the following fields that provide details about a thread that was selected for cancelation and for which the post-cancel exit was called. This information is provided for CHECK function calls only.

UXPO_CONNECT_TYPE
(CONNECTION TYPE). Specifies the numeric identifier for the connection type. A connection type can be a system region or program from which the thread originated or the type of distributed access protocols that the thread is using. The name of the connection type in character format is provided in the CONNECTION TYPE CHAR field.

UXPO_THREAD_TOKEN
(THREAD TOKEN). Specifies the thread token value for the thread. A thread token value is a unique numeric identifier that DB2 assigns to each active thread on a DB2 subsystem.

UXPO_ASID_BINARY
(BINARY ASID). Specifies the address space identifier (in binary format) for the address space from which the thread originated.

UXPO_SQL_REQ_COUNT
(SQL REQUEST COUNT). Specifies the number of SQL calls that the thread performed.

UXPO_ACTIVE_IN_DB2
(ACTIVE IN DB2). Specifies the thread status:

- In-DB2: The thread is active and running in DB2.
- In-APPL: The thread is not active in DB2.
- DEF-TERM: The thread is active and in deferred termination processing.
- Q-DEFTERM: The thread is active and is queued for deferred termination processing.

UXPO_STATUS
(THREAD STATUS). Specifies the DB2 connection status code for the thread. Many codes are available and are described in the message DSNV404I in the *DB2 Version 9.1 for z/OS Messages* manual.

UXPO_ASID
(ASID). Specifies the address space identifier (in character hex format) for the address space from which the thread originated.

UXPO_IP_PORT
(IP PORT). Specifies the IP port number of the Distributed Database Facility (DDF) requestor from which the threads originated.

UXPO_JOBID
(JOBID). Specifies the unique identifier that the z/OS system generated for the job from which the thread originated.

UXPO_AUTH_ID
(AUTH ID). Specifies the current authorization ID of the process that is associated with the thread. An authorization ID is a character string that can be verified for connection to DB2. If the SET CURRENT SQLID statement was used to change the original authorization ID of the process, this field displays the ID that is currently used and the **ORIGINAL AUTH ID** field displays the authorization ID that was initially used.

UXPO_PLAN
(PLAN). Specifies the name of the DB2 plan under which the thread is running.

UXPO_MEMBER_NAME
(MEMBER NAME). Specifies the data-sharing member name of the subsystem that the thread is referencing within a data-sharing group.

UXPO_ORIG_AUTH_ID
(ORIGINAL AUTH ID). Specifies the original authorization identifier of the process for which the threads were initially established.

UXPO_CONN_ID
(CONNECTION TYPE CHAR). Specifies the connection type in character format. A connection type can be a system region or program from which the thread originated or the type of distributed access protocols that the thread is using.

- CAF: The thread originates from a Call Attach Facility (CAF) user.
- CICS: The thread originates from a CICS region.
- DBAT_DRD: The thread is a distributed database access thread that uses distributed relational database access protocol (DRDA).
- DBAT_PP: The thread is a distributed database access thread that uses distributed database private protocols (DDF).
- IMSBMP: The thread originates from an IMS Batch Message Processing (BMP) program.
- IMSCTL: The thread originates from an IMS control region.
- IMSDLIBT: The thread originates from an IMS Data Language/I (DL/I) batch region.
- IMSMPP: The thread originates from an IMS message region.
- IMSTBMP: The thread originates from an IMS transaction BMP.
- RRSAF: The thread originates from the Recoverable Resource Manager Service attachment facility.

- TSO: The thread originates from a TSO/E interactive or batch user.
- UTILITY: The thread originates from a DB2 utility.

UXPO_PROGRAM_NAME
(PROGRAM NAME). Specifies the name of the program that is currently running with the thread.

UXPO_JOBNAME
(JOBNAME). Specifies the name of the job from which the thread originated. This name is specified in the job card.

UXPO_NAME
(CONNECTION NAME). Specifies the name of the connection. This name is the same as the **Name** value that is returned from the DB2 -DISPLAY THREAD command.

UXPO_CORR_ID
(CORRELATION ID). Specifies the correlation ID of the thread. A correlation ID is an identifier that is associated with a specific thread. For example, it can be a TSO user ID or the job name.

UXPO_IP_ADDRESS
(IP ADDRESS). Specifies the IP address of the DDF requestor from which the thread originated. An IP address is a unique address of a location on the Internet.

UXPO_LOCATION_NAME
(LOCATION NAME). Specifies the name of the remote DDF requestor location from which the thread originated.

UXPO_COLLECTION_ID
(COLLECTION NAME). Specifies the collection identifier for the group of packages that includes the package under which the thread was running.

UXPO_PACKAGE_NAME
(PACKAGE NAME). Specifies the name of the DB2 package under which the thread is running.

Supported wildcard characters

DB2 UET supports the standard DB2 wildcard characters.

These wildcard characters are:
- The percent sign (%) for zero or more characters
- The underscore (_) for a single character

You can include wildcards in most of the thread-filtering criteria that you can specify from the ISPF interface or batch interface. You can also include wildcards in the attribute values for some elements in the DSNUTILB intercept policy.

DB2 UET matches any pattern of wildcards and characters that you specify against the data that DB2 maintains. For example, if you specify PAY% for the name of a database for a thread-cancelation operation, DB2 UET selects all threads that are referencing any database that has a name beginning with "PAY."

You can also specify a wildcard escape character on the Set Personal Defaults panel in the DB2 UET ISPF interface or by using the ESCAPE parameter that the batch interface and DSNUTILB intercept support. An escape character delimits a

wildcard character in a parameter value when that wildcard character is used as a part of the data value rather than as a wildcard.

For more information about DB2 wildcards and escape characters, see the section on LIKE predicates in the *DB2 for z/OS SQL Reference*.

Notices

This information was developed for products and services offered in the U.S.A.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.html.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the Unites States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and the section titled "Cookies, Web Beacons, and Other Technologies" in IBM's Online Privacy Statement at http://www.ibm.com/privacy/details. Also, see the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Product Number: 5655-T58

Printed in USA

SC19-3417-05