Tivoli. IBM Tivoli Asset Discovery for z/OS
Version 7.2

IBM

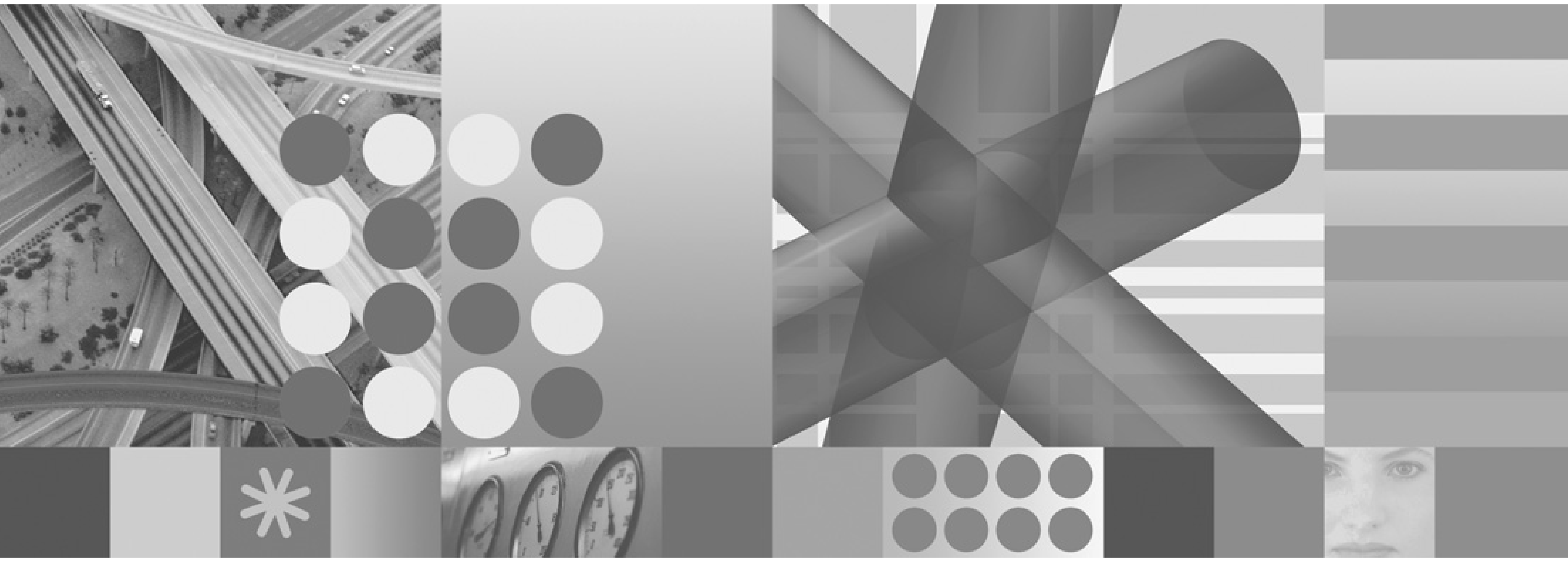
Version 7 Release 2

Administration Guide and Reference

SC27-2701-00

Tivoli® IBM Tivoli Asset Discovery for z/OS
Version 7.2

Version 7 Release 2



Administration Guide and Reference

SC27-2701-00

Note

Before using this information and the product it supports, read the information in “Notices” on page 207.

First edition (August 2009)

This edition applies to Tivoli Asset Discovery for z/OS, Version 7, Release 2 (product number 5698-B39) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

About this document

The IBM® Tivoli® Asset Discovery for z/OS® Administration Guide and Reference explains how to install, configure, and use the product.

Who should read this document

This document is intended for system administrators who install and configure Tivoli Asset Discovery for z/OS, or who perform daily system management tasks, such as inventory gathering and reporting of software distribution.

What this document contains

- Chapters 1 - 6

  Provide information on installation and migration processes and are a reference guide to the product.
- Chapters 7 - 17

  Provide general reference information.
- Appendix A, "Messages," on page 135

  Provides a reference for the messages.
- Appendix B, "Repository table layouts," on page 191

  Provides a reference for the Repository table layouts.
- Appendix C, "Deletion of obsolete libraries in Dorana Version 5 Release 4 Repository," on page 205

  Provides a reference for the housekeeping of some obsolete libraries in Dorana Version 5 Release 4.

How to read the syntax diagrams

The syntactical structure of commands described in this document is shown by means of syntax diagrams.

Figure 1 shows a sample syntax diagram that includes the various notations used to indicate such things as whether:
- An item is a keyword or a variable.
- An item is required or optional.
- A choice is available.
- A default applies if you do not specify a value.
- You can repeat an item.

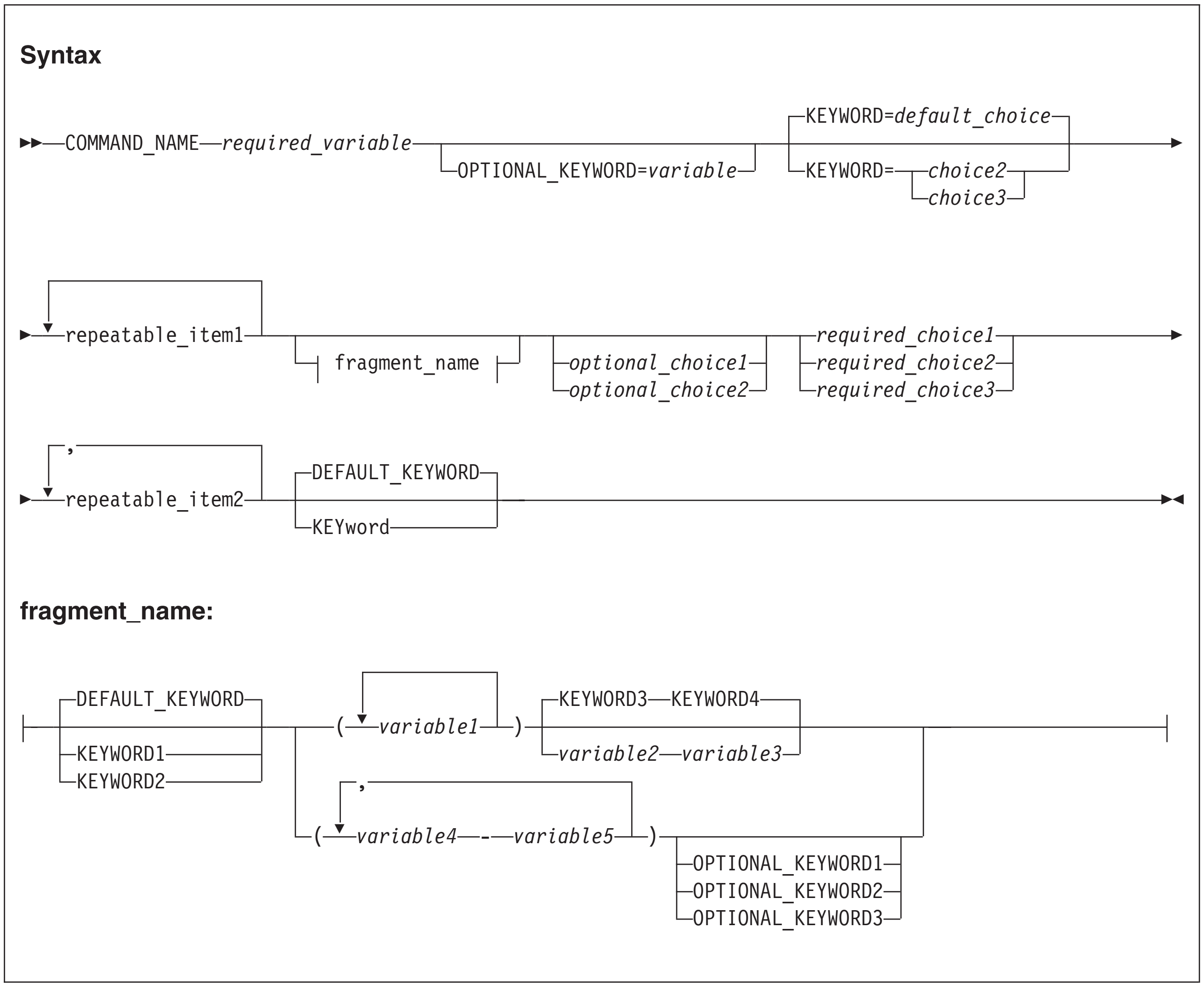


Figure 1. Sample syntax diagram

Here are some tips for reading and understanding syntax diagrams:

Order of reading

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The ▶▶─── symbol indicates the beginning of a statement.

The ───▶ symbol indicates that a statement is continued on the next line.

The ▶─── symbol indicates that a statement is continued from the previous line.

The ───▶◀ symbol indicates the end of a statement.

Keywords

Keywords appear in uppercase letters.

```
▶▶──COMMAND_NAME──────────────────────────────────────────────────▶◀
```

Sometimes you only need to type the first few letters of a keyword, The required part of the keyword appears in uppercase letters.

```
     ┌─DEFAULT_KEYWORD─┐
▶▶───┤                 ├──────────────────────────────────────────▶◀
     └─KEYword─────────┘
```

In this example, you could type ″KEY″, ″KEYW″, ″KEYWO″, ″KEYWOR″ or ″KEYWORD″.

The abbreviated or whole keyword you enter must be spelled exactly as shown.

Variables

Variables appear in lowercase letters. They represent user-supplied names or values.

```
▶▶──required_variable─────────────────────────────────────────────▶◀
```

Required items

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

```
▶▶──COMMAND_NAME──required_variable───────────────────────────────▶◀
```

Optional items

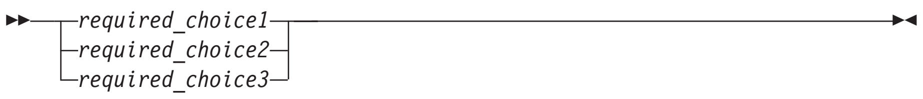
Optional items appear below the main path.

```
▶▶───┬──────────────────────────────┬─────────────────────────────▶◀
     └─OPTIONAL_KEYWORD=variable────┘
```

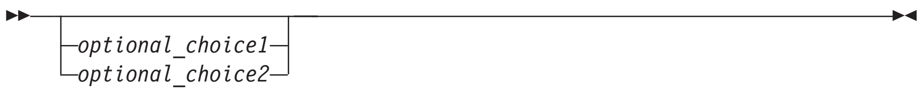
Choice of items

If you can choose from two or more items, they appear vertically, in a stack.

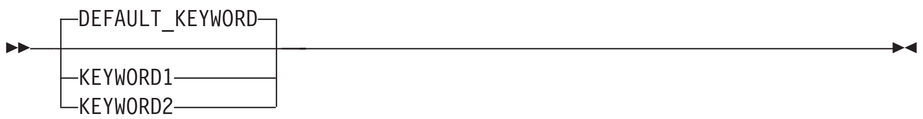
If you *must* choose one of the items, one item of the stack appears on the main path.



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

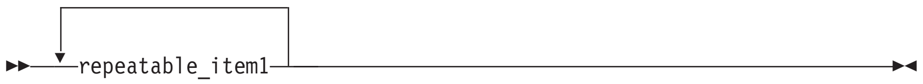


If a default value applies when you do not choose any of the items, the default value appears above the main path.

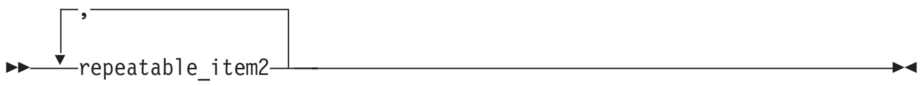


Repeatable items

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

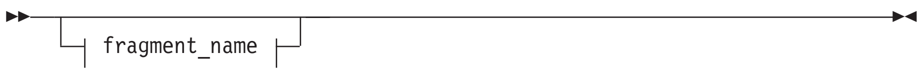


If you need to specify a separator character (such as a comma) between repeatable items, the line with the arrow returning to the left shows the separator character you must specify.



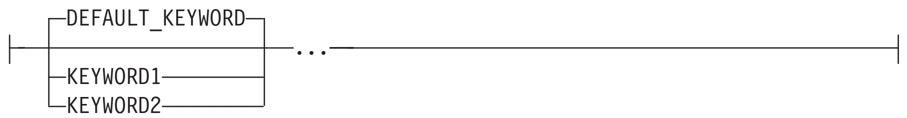
Fragments

Where it makes the syntax diagram easier to read, a section or *fragment* of the syntax is sometimes shown separately.



.
.
.

fragment_name:



Chapter 1. Overview of IBM Tivoli Asset Discovery for z/OS

The Tivoli Asset Discovery for z/OS runs on z/Architecture® mainframes executing the z/OS operating system. Its purpose is to:

- Discover and identify services for the z/OS platform.
- Monitor software usage and trends.
- Report on the MSU capacity of each LPAR under which the product runs.
- Provide reporting for assets and usage.

The benefits of using this software are:

- Used and unused software are identified.
- Users of software are identified.
- Obsolete versions of software are identified and the usage of these versions determined.
- Usage trends of software and libraries are identified.

In an IBM z/OS environment, software is contained within load libraries as load modules, also known as LMODS, or program objects in the z/OS UNIX® file systems. The software products installed, and usage of those products, is accurately determined by scanning the content of these libraries and files. The product contains a number of components designed to carry out this task. In addition, the scanned load libraries, load modules, and z/OS UNIX files are checked against a database of product information supplied by this software. This database allows the product to infer, from the contents of the load libraries and z/OS UNIX files, what products are installed on a system, or made available to the system to use. The database component is known in the product as the Global Knowledge Base, or GKB. In addition there is a Local Knowledge Base, or LKB, in which you can define your own software or occasional changes to the GKB. In specific circumstances, the LKB information is used in preference to the GKB.

Throughout this document, the concept of a schema is used. Following is a brief description of this concept as used in DB2. A schema provides a logical classification of objects in the database. The objects that a schema can contain include tables, indexes, table spaces, distinct types, functions, stored procedures, and triggers. An object is assigned to a schema when it is created. When a table or index is created, it is given a qualified two-part name. The first part is the schema name (or the qualifier) and the second part is the name of the table or index.

This diagram shows the product workflow.

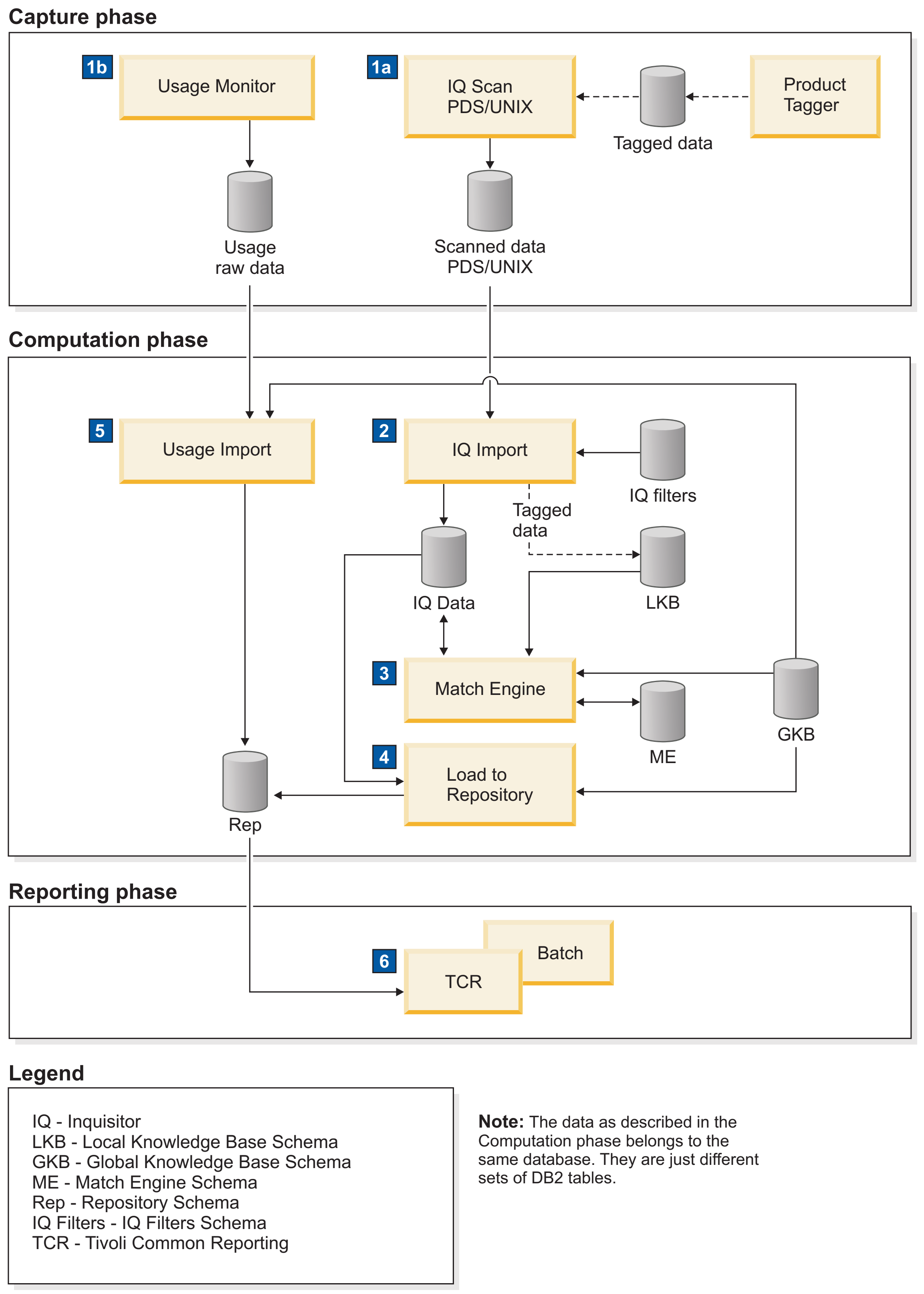


Figure 2. Tivoli Asset Discovery for z/OS workflow

Inquisitor
A set of programs which find loadable programs in z/OS data sets and UNIX System Services file systems. One program locates load libraries on

z/OS DASD devices, opens these load libraries, and captures information from the load modules located in the libraries. This program is targeted to specific devices, libraries, or groups of libraries. It creates a compressed data set, which is then used as input to the Inquisitor Import process.

Another program locates and scans z/OS UNIX directories for program objects, and captures this information. It creates a compressed data set, which is then used as input to the Inquisitor Import for z/OS UNIX processing.

Product Tagger
A utility that allows the identification of software which has not been predefined in the Global Knowledge Database. It can also be used to supersede GKB entries without changing the GKB contents.

Usage Monitor
A program which attaches to the z/OS program management functions, using standard IBM attachment protocols. It is able to monitor the execution of load modules and to record these events.

Inquisitor Import
A set of programs that load raw Inquisitor data into IBM DB2® tables on z/OS, for z/OS load modules and z/OS UNIX program objects.

Match Engine
The process whereby the imported Inquisitor data is matched against a Global Knowledge Database. This database is provided by IBM. The Local Knowledge Database, populated by the user, can also be used during the match process.

Load to Repository
The process whereby the matched Inquisitor data is copied into the Repository. The Repository is where all data resides.

Usage Import
The process whereby the raw Usage Monitor data is copied into the Repository against matching load modules. Once completed, the user views the Usage data from the IBM Tivoli Common Reporting reports, or by submitting batch reports.

IBM Tivoli Common Reporting and Batch Reporting
IBM Tivoli Common Reporting and Batch Reporting, provide the basis for reporting the results of the scanned systems. The product provides several reports that are accessed from Tivoli Common Reporting or Batch. These reports determine what and where the products are located on the system.

Data Capture phase tasks

The Data Capture phase consists of running the Inquisitor and the Usage Monitor on the target system, or systems, that the information is required from. The Data Capture phase does not require DB2. The data sets created by Data Capture are then used by the Computation phase.

Running the Inquisitor

For each of the schemas specified, submit the appropriate job to scan for information. If you have NJE, you can submit the jobs from a single system. It is important that the job runs on the appropriate system. For sysplex or shared system schemas, you only have to run one job for each sysplex or shared system. The job can run on any member of the sysplex, or any system, in the shared systems.

Running the Usage Monitor

The Usage Monitor can be started depending on how you have deployed your system. If you have NJE, you can submit the jobs from a single system. However, make sure that the job runs on the appropriate system. If the schema is for a sysplex or shared system, you can run the same job on every member of the sysplex or on each shared system. If the systems are in a Multi-Access Spool environment, you might have to change the job name in each system in order to get the jobs to run.

The monitor runs continuously, capturing data on what programs the system is running. This information is written to a data space. The data is available for further processing whenever it is off-loaded to a data set from the data space. This occurs by default at midnight, or you can set a switch to force the offload to occur.

Computation phase tasks

The Computation phase consists of importing the data captured by the Data Capture phase into DB2 tables, and then matching that captured data with the information in the GKB and LKB. The result is a database of identified installed products with optional usage information. The result of the matched imported data is then loaded to the Repository where the Reporting phase can use it.

The computation phase tasks all run on the DB2 system. They involve:
1. Importing the Inquisitor captured data.
2. Matching the captured data with known product information from the Global and Local Knowledge bases.
3. Loading the matched information into the Repository.
4. Importing the Usage captured data into the Repository.
5. Running Utilities to manage and maintain your data. These tasks are optional.

Reporting phase tasks

The report phase tasks all run against the DB2 system. They involve:
1. Using Tivoli Common Reporting to view your data.
2. Running batch SQL queries to view your data.

Chapter 2. Deployment scenarios

When deploying Tivoli Asset Discovery for z/OS, there are some concepts you need to understand in order to comprehend the data and correctly utilize the product.

Inquisitor data
The Inquisitor scans DASD volumes for libraries containing load modules and HFS/zFS volumes for z/OS UNIX program objects. The Match Engine associates these libraries and load modules with a Vendor, Product, Option, and Release. The Load to Repository then loads this information into the DB2 Repository. This group of information forms the basis for an Inventory.

Usage event
A Usage event describes a unique load of a load module, or UNIX program object, and can contain extra details such as job name, username, and account code. The Usage Monitor records these Usage events as they occur on a particular operating system. When the Usage data is imported into the Repository, an SMF ID or sysplex name on which the load module was loaded, is associated with an Inventory. After the Usage data is loaded into the Repository, each Usage event is identified by the load module name, data set name, and volume.

Region
A Region is a logical group of inventories.

The diagram shown here illustrates the hierarchical view of the product concepts.

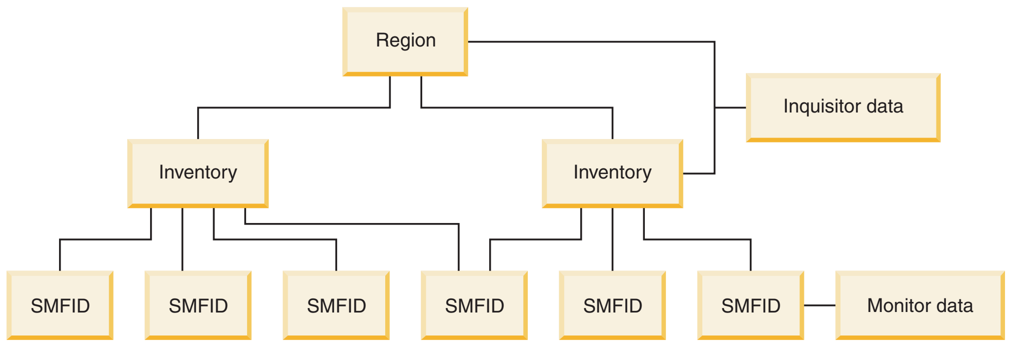


Figure 3. Hierarchical view of Inventory association

An Inventory consists of:

- All the load modules, load libraries, and z/OS UNIX program objects that exist on a set of scanned DASD.
- The products that these load modules and z/OS UNIX program objects are associated with.
- The Usage events, for these load modules and z/OS UNIX program objects, for each z/OS system that is being monitored.

It utilizes the concepts mentioned above for each Inventory to run in three distinct modes. A Repository can have any number of Inventories running in any combination of modes.

Single or stand-alone mode
: Single or stand-alone mode Inventory represents a single z/OS system with access to a set of nonshared DASD. For example, if you have one z/OS system that has access to a set of nonshared DASD, then you would run the resulting Inventory in stand-alone mode. There would be an Inquisitor and a Usage Monitor running on the single z/OS system.

Sysplex mode
: A sysplex mode Inventory represents a group of z/OS systems running in a sysplex configuration. There would be one z/OS system running the Inquisitor, and all systems running their own Usage Monitor. All Inquisitor and Usage Monitors would need to be run in sysplex mode, and Usage data would be matched to an Inventory using the sysplex name instead of the SMF ID.

Shared DASD mode
: A Shared DASD mode Inventory represents a group of z/OS systems that have a shared pool of DASD, but are not in a sysplex environment, and do not have a common sysplex name. As in sysplex mode, a single system can run the Inquisitor scan on behalf of all the other systems, and all systems would also run their own Usage Monitor. However, the Usage data would be matched to the Inventory based upon SMF ID, instead of the sysplex name. For more information, refer to “Add Inventory Association” on page 121.

It is recommended to have a central DB2 subsystem that contains the Repository of the entire enterprise. The DB2 subsystem that contains this Repository, and which the Computation phase is run on, does not need to reside on the same systems that you are collecting information from. The Usage and Inquisitor data that require processing can be transmitted to this centralized DB2 subsystem using the Tivoli Asset Discovery for z/OS Automation server or equivalent automation product.

In all the examples shown here, no centralized DB2 subsystem is displayed, because there will be one for the entire enterprise, and this can reside on any system.

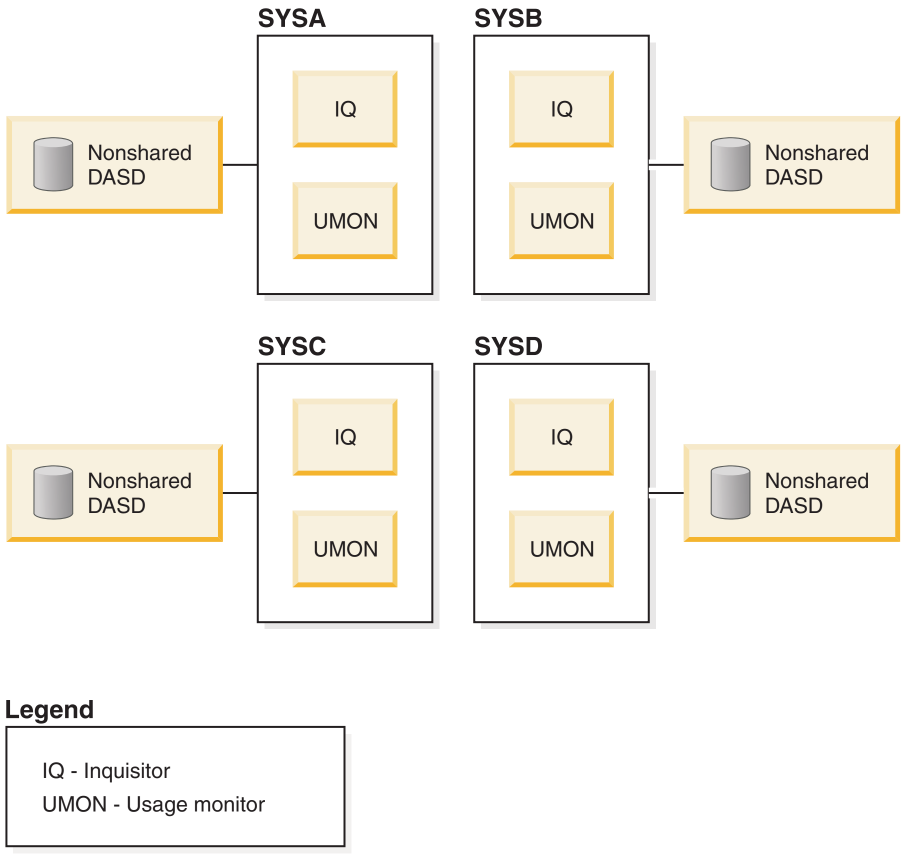


Figure 4. Four LPARS, each with own set of DASD

In this example there are four Inventories, with one SMF ID associated to each Inventory. The Usage Monitor and the Inquisitor run on all four z/OS systems and DASD, respectively. In this case all four Inventories are run in the single or stand-alone mode

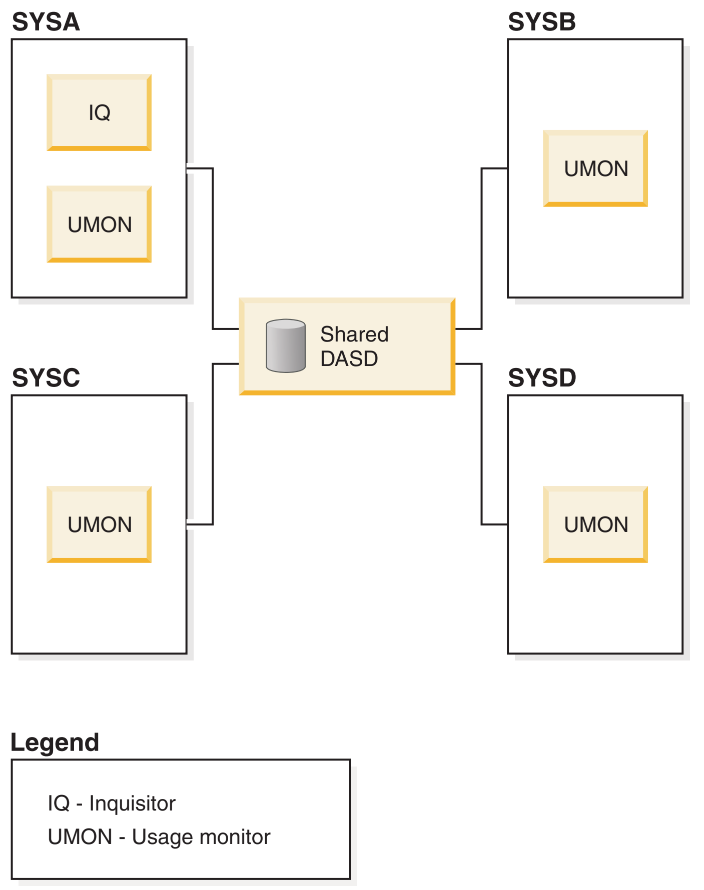


Figure 5. Four LPARs, all with Shared DASD, Non-Sysplex

In this example, there is one Inventory, the shared DASD, and four SMF IDs that are linked to that Inventory. The Usage Monitor runs on all four z/OS systems, but there is only a need for one Inquisitor, as all the z/OS systems have access to the same load modules. In this case the Inventory is in the Shared DASD mode, as the z/OS systems are not in a Sysplex configuration.

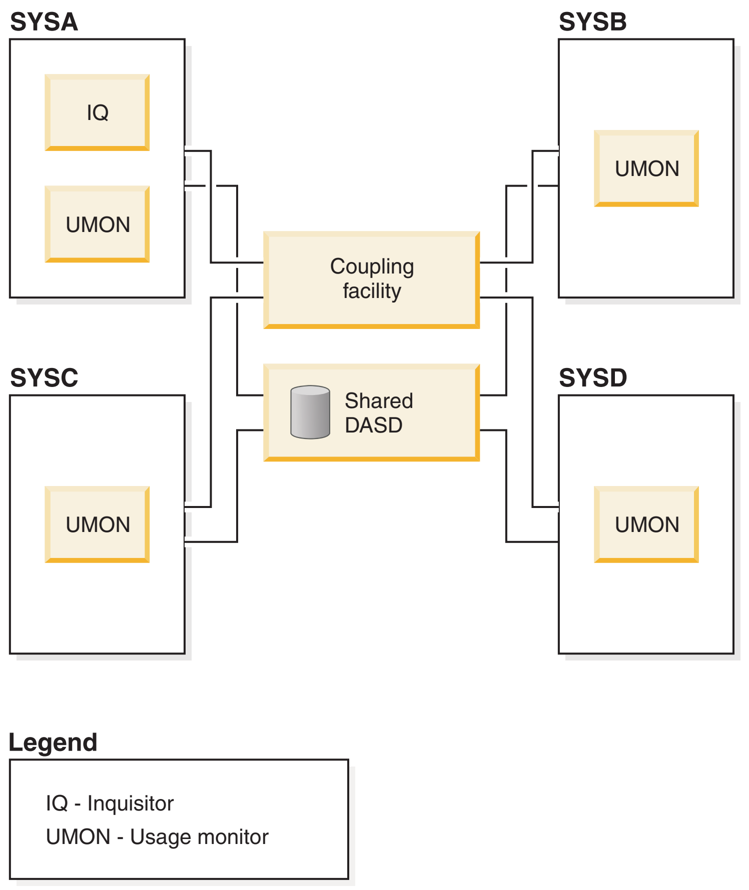


Figure 6. Sysplex of four LPARs all with Shared DASD

In this example there is one Inventory, the shared DASD, and as this is a sysplex configuration, the Usage data is linked to the Inventory through the SYSPLEX name. The Usage Monitors and the Inquisitor are run in PLX=Y mode. Only one Inquisitor needs to be run on the Shared DASD. The Usage Monitor needs to be run on each z/OS system. In this case the Inventory is run in the Sysplex mode

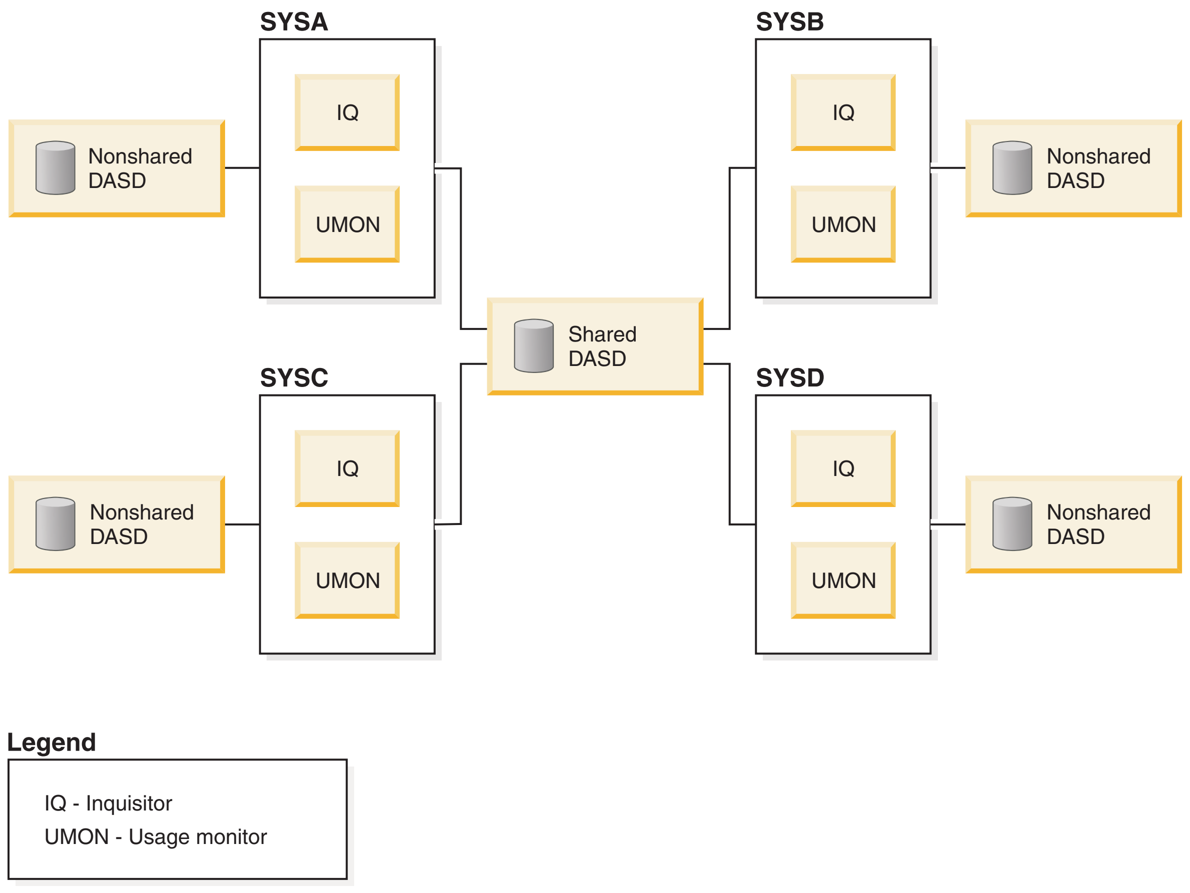


Figure 7. Four LPARs, one set of shared DASD, plus a set of nonshared DASD for each LPAR

This diagram shows four z/OS systems, all with access to shared DASD, as well as a set of non shared DASD. There will be five Inventories, one for the shared DASD, and one for each of the nonshared DASD. Each z/OS system would be associated with the shared DASD Inventory through the SMF ID, and also with the nonshared DASD, in a single or stand-alone mode, also through the SMF ID. The Usage Monitor and the Inquisitor both run on all four systems. Three of these systems need to run the Inquisitor against just the nonshared DASD, and one system needs to run the Inquisitor against the shared and its nonshared DASD. In this case there is one shared DASD mode Inventory and four single or stand-alone mode Inventories.

Chapter 3. Installation requirements

Hardware requirements

The basic hardware requirements to install Tivoli Asset Discovery for z/OS are:

Processors

Data Capture and Computation phases
These phases require a z/Architecture machine capable of running z/OS Version 1 Release 8 or later.

Reporting phase
The Batch Reports require a z/Architecture machine capable of running z/OS Version 1 Release 8 or later. The Tivoli Common Reporting systems require either an Intel® x86 architecture server, or Linux® on System z® LPARs, in which case a z/Architecture machine is used . For best performance, processor speeds should be at least 1 GHz for RISC architectures and 2 GHz for Intel® architectures.

In addition Tivoli Common Reporting requires the following:

- Process memory requirement - 2 GB
- Disk storage requirement - 662 MB
- As a prerequisite installation for Java Runtime Environment (JRE), download the Red Hat Enterprise Linux® and SUSE: compat-libstdc++-33-3.2.3-47.3 package.

Software requirements

The software requirements for running Tivoli Asset Discovery for z/OS are:

z/Architecture software

Installation	• z/OS Version 1 Release 8 or later • SMP/E Version 3 Release 3 or later • DFSMSDSS Version 1 Release 8 or later
Data Capture phase	• z/OS Version 1 Release 8 or later
Computation phase	• z/OS Version 1 Release 8 or later • Language Environment® for z/OS • DB2 Version 8 Release 1 (new function mode) or later • DDF enabled for DB2 • CLI (ODBC) enabled for DB2 • REXX™ enabled for DB2

Reporting phase	• z/OS Version 1 Release 8 or later • DB2 Version 8 Release 1 (new function mode) or later • DDF enabled for DB2 • CLI (ODBC) enabled for DB2 • REXX enabled for DB2 If the Tivoli Common Reporting component is to run on Linux on System z: • Linux on System z (SUSE Version 10 or later)

Tivoli Common Reporting Software

Operating system:	Supported operating systems are 32-bit only: • Solaris Version 9, 10 or 11 • Red Hat Enterprise Linux Version 4 or 5 • Red Hat Enterprise Linux 5 on System z • SUSE Linux Version 9 or 10 • SUSE Linux 10 on System z • HP-UX Version 11iv2 or 11iv3 • IBM AIX® Version 5 Release 3 or Version 6 Release 1 • Microsoft® Windows® 2003 Server • Microsoft Windows XP • Microsoft Windows 2008
Reporting Tool	Tivoli Common Reporting Version 1 Release 2, or later Supported browsers are Internet Explorer and Firefox
Database Communications	DB2® Connect™ Personal Edition Version 8 Release 1, or Version 9 Release 1 DB2 Connect Enterprise Edition Version 8 Release 1, or Server Version 9 Release 1

Security and authorization

TSO user ID

You require a TSO user ID with the appropriate RACF® access to run jobs in the Data Capture phase, Computation phase, and Reporting phase.

DB2 authorization IDs

You need DB2 privileges to perform the following tasks:

- DBADM authority to access the product database. You need to drop and create DB2 resources.
- BIND plans and packages.
- EXECUTE authority to execute plans and packages.
- SELECT authority to access the product tables.
- DISPLAY authority to issue the DISPLAY DDF command.

- LOAD and STATS privileges to run DB2 utilities LOAD and RUNSTATS.
- Access to work file database or TEMP database for ″Declared Global Temporary table″.

APF

It is recommended that load library SHISMOD1 be APF authorized. This is required for running the Inquisitor scan and Usage Monitor.

z/OS UNIX security

For more information refer to “Security considerations when running the Inquisitor against z/OS UNIX files” on page 64.

Administrator authority for Tivoli Common Reporting server

You require Administrator authority to install Tivoli Common Reporting on the server.

Chapter 4. Installation

Installing Tivoli Asset Discovery for z/OS

See the *Program Directory* for details of how to install Tivoli Asset Discovery for z/OS.

Installing Tivoli Common Reporting

Installation of Tivoli Common Reporting depends on where the tool is to be installed. Tivoli Common Reporting can be installed on many different operating environments.

See the *IBM Tivoli Common Reporting User's Guide* for details of the process required to install Tivoli Common Reporting.

Chapter 5. Post-installation customization

After you have installed Tivoli Asset Discovery for z/OS, there are some customization tasks that need to be completed.

z/OS Customization

Tivoli Asset Discovery for z/OS requires some post-installation tasks to enable the product to run efficiently in the z/OS environment.

APF Authorization

Some of the product's components use z/OS authorized system services and can only be run from an APF authorized library. It is recommended that the load library SHSIMOD1 is APF authorized. Alternatively, the modules listed here can be copied to an APF authorized library.

- The Monitor module HSIZMON can only be run from an APF library.
- The Inquisitor module HSIPINQ, by default, requires APF.

  However, it can be run from a non-APF library when using the NOAPF parameter. When using NOAPF mode, the Inquisitor bypasses the use of z/OS authorized system services and is only able to collect a subset of the data. APF mode is recommended for scanning accuracy and speed.
- All other modules can be run from an APF library or non-APF library.

MAXCAD parameter

The Usage Monitor uses a SCOPE=COMMON data space for its repository. For this reason, it is necessary to have available at least two additional system wide data space PASN entries. Tivoli Asset Discovery for z/OS uses one data space, but after a switch, a new one is created. The older data space is not deleted until it has been processed by the writer task.

To allow the creation of the Usage Monitor data spaces, the MAXCAD system parameter must be set to a large enough value. The MAXCAD parameter, if specified, is located in the IEASYSxx member of the system's PARMLIB library. Consult the relevant MVS™ Initialization and Tuning Reference manual for the default and valid value range. The default value must be considered as the minimum value for practical purposes. Other products also make use of this facility, and the MAXCAD value used must take into account all the various product requirements for this facility.

DB2 customization

Tivoli Asset Discovery for z/OS requires post-installation tasks to enable DB2 functions and define DB2 resources for the product to use. An authorized database personnel with the appropriate authorities needs to perform these tasks:

- Bind Call level interface/ODBC support

  The product uses the DB2 supplied DSNACLI plan. Bind the DSNACLI plan using the job 'hlq.SDSNSAMP(DSNTIJCL)'. In order to avoid a common error associated with SQL error code of -805 (DBRM OR PACKAGE NAME NOT FOUND IN PLAN), it is recommended that you rebind this plan with the latest DB2 maintenance.
- Bind REXX language support

uses the DB2 supplied DSNREXX plan. Bind the DSNREXX plan using the job 'hlq.SDSNSAMP(DSNTIJRX)'..

- DB2 accesses

  It is necessary for the product administrator to have DBADM access to the DB2 database, as the computation phase utilities require create and drop access to DB2 resources.

  For Tivoli Common Reporting and Batch Reporting accesses to DB2 resources, the definitions are found in HSISGRNT, GI*, and GX*. For more information, see "Creating post-installation jobs."

Installing DB2 Connect (if required)

DB2 driver and license JARs are required to create a JDBC connection between Tivoli Common Reporting and the Tivoli Asset Discovery for z/OS Repository. Installing DB2 Connect provides these JARs. DB2 Connect is part of the "DB2 Management Clients Package" when you purchase DB2 for z/OS.

Note: This product is to be installed on a Windows or UNIX platform.

If DB2 Connect is already installed, a separate instance of this product is not required. If not, install DB2 Connect using these steps:

1. On the installation media select and run the DB2 Connect **setup.exe**
2. Select **Install a Product** from the left menu.
3. Select **Install Now** to begin the DB2 Connect installation.
4. Follow the default options in the wizard to install DB2 Connect.
5. Once the DB2 First Steps wizard appears, select **Exit** from the left menu to complete the installation.
6. Bind the DB2 CLI bind files to DB2 on z/OS:
   a. Ensure you connect to the DB2 subsystem on z/OS
   b. Run the bind process either by the Configuration Assistant (CA), or issue `bind @db2cli.lst messages` from the Command Processor.

Creating post-installation jobs

In order to create a customized version of the post-installation JCLLIB and PARMLIB, take a copy of the job in member HSISCUST in the 'hlq'.SHSISAMP data set, and edit it. This job contains parameters to define DB2 objects, and jobs to run the product on the system. It is recommended that you seek the assistance of a DB2 database administrator in customizing DB2-related parameters, and a system programmer in customizing SMS-related parameters.

Change the following parameters in HSISCUST as required:

Parameter	Description
SET HSI	This JCL parameter must be set to the high level qualifiers of the target libraries created by the SMP/E installation process. These data sets all have low level qualifiers beginning with SHSI.

SET ISP	The customization tool uses ISPF services to customize the parameters and JCL for the user. This parameter specifies the high level qualifiers for the ISPF target libraries. These libraries all have low level qualifiers beginning with SISP.
MODE	The parameter is set in the PARM option of the EXEC statement. **MODE=POST** Members beginning with HSIS are replaced. They are not related to IQ schemas. **MODE=IQ** Members not beginning with HSIS are replaced. These are related to IQ schemas. **MODE=ALL** All members are replaced. No members are ever deleted. Run with MODE=ALL for the first time after installation.
HSIINST	The HSIINST parameter is in the //SYSIN part of the job. This parameter specifies the high level qualifiers of the JCLLIB and PARMLIB data sets that are created by running the HSISCUST job. If the JCLLIB and PARMLIB data sets already exist, they are reused and members can be replaced with updated information. The name specified for this parameter must be less than or equal to 18 characters in length.
DB2LOAD	The DB2LOAD parameter is in the //SYSIN part of the job. This parameter specifies the fully qualified SDSNLOAD data set name.
DB2EXIT	The DB2EXIT parameter is in the //SYSIN part of the job. This parameter specifies the fully qualified SDSNEXIT data set name. If the DB2EXIT library does not exist, use the same value as the DB2LOAD parameter.
DB2RUN	The DB2RUN parameter is in the //SYSIN part of the job. This parameter specifies the fully qualified RUNLIB data set name.
CEERUN	The CEERUN parameter is in the //SYSIN part of the job. This parameter specifies the fully qualified Language Environment data set.
CBCDLL	The CBCDLL parameter is in the //SYSIN part of the job. This parameter specifies the fully qualified Language Environment C++ runtime data set.
DBSSID	The DBSSID parameter is in the //SYSIN part of the job. This parameter specifies the DB2 subsystem ID on the z/OS system.

LOC	The LOC parameter is in the //SYSIN part of the job. This parameter specifies the ODBC(CLI) location for the DB2 subsystem ID on the z/OS system. You can use the DB2 DISPLAY DDF command to determine the location.
TIADPLAN	The TIADPLAN parameter is in the //SYSIN part of the job. This parameter specifies the name of the DB2 plan used by the DSNTIAD utility. This utility allows for the execution of dynamic SQL in a batch job and is required by the post-installation jobs. The name of this plan is obtained from the DB2 database administrator.
SGTABCAT	The SGTABCAT parameter is in the //SYSIN part of the job. This parameter specifies the prefix of the DB2 table space data set names for small tables in the database. Consult your DB2 database administrator for security implications and disk storage requirements. See the SQL statement CREATE STOGROUP for more information. This parameter specifies the storage group name. It is compulsory to use the storage group name, SGHSITAB, as supplied with this product.
SGTABVOL	The SGTABVOL parameter is in the //SYSIN part of the job. This parameter specifies the names of the volumes that the table space data sets, for small tables, are allocated on. See the SQL statement CREATE STOGROUP for more information. This parameter specifies the volumes.
SGBIGCAT	The SGBIGCAT parameter is in the //SYSIN part of the job. This parameter specifies the prefix of the DB2 table space data set names for large tables in the database. Consult your DB2 database administrator for security implications and disk storage requirements. See the SQL statement CREATE STOGROUP for more information. This parameter specifies the storage group name. This parameter specifies the storage group name. It is compulsory to use the storage group name, SGHSIBIG, as supplied with this product.
SGBIGVOL	The SGBIGVOL parameter is in the //SYSIN part of the job. This parameter specifies the names of the volumes that the table space data sets, for large tables, are allocated on. See the SQL statement CREATE STOGROUP for more information. This parameter specifies the volumes.

SGIDXCAT	The SGIDXCAT parameter is in the //SYSIN part of the job. This parameter specifies the prefix of the DB2 table space data set names for table indexes in the database. Consult your DB2 database administrator for security implications and disk storage requirements. See the SQL statement CREATE STOGROUP for more information. This parameter specifies the storage group name. This parameter specifies the storage group name. It is compulsory to use the storage group name, SGHSIIDX, as supplied with this product.
SGIDXVOL	The SGIDXVOL parameter is in the //SYSIN part of the job. This parameter specifies the names of the volumes that the table space data sets, for table indexes, are allocated on. See the SQL statement CREATE STOGROUP for more information. This parameter specifies the volumes.
DB	The DB parameter is in the //SYSIN part of the job. This parameter specifies the name of the DB2 database that the product uses to store all the information that it gathers. The DB name must be less than or equal to 8 characters in length.
DBADMIN	The DBADMIN parameter is in the //SYSIN part of the job. This parameter specifies the list of user IDs that are to be granted ADMIN access to the database and its contents. This parameter is optional. Specify an empty string if you do not want to grant ADMIN access to user IDs for the database specified in DB.
KBMGMTC	The KBMGMTC parameter is in the //SYSIN part of the job. This parameter specifies SMS management class for allocation of the data sets created when loading the Global Knowledge Base (and z/OS UNIX GKB). This class name is optional but the parameter must be specified. Specify a null ('') if it is not used. The KBMGMTC and KBSTORC parameters are used to specify where the data sets are to be allocated. If SMS is not enabled, or you wish to specify volumes to be used, specify nulls for both KBMGMTC and KBSTORC and specify the volume names in KBVOLS.
KBSTORC	The KBSTORC parameter is in the //SYSIN part of the job. This parameter specifies SMS storage class for allocation of the data sets created when loading the Global Knowledge Base (and z/OS UNIX GKB). This class name is optional, but the parameter must be specified. Specify a null ('') if it is not used. See the KBMGMTC entry for details.

KBVOLS	The KBVOLS parameter is in the //SYSIN part of the job. This parameter specifies DASD volumes to allocate the data sets used in the loading of the Global Knowledge Base (and z/OS UNIX GKB). The volumes specification is optional, but the parameter must be specified. Specify a null ('') if it is not used. The format of data in this parameter is '(volume) [,(volume)]...'. This parameter is placed in the DFSMSDSS RESTORE command OUTDYNAM() parameter. See the KBMGMTC entry for details. You must specify either KBMGMTC or KBSTORC, or KBVOLS
KBCPYPND	The KBCPYPND parameter is in the //SYSIN part of the job. This parameter specifies whether to run a DB2 utility to turn the COPY PENDING flag off for Knowledge Base Schemas after reloading the schema tables with the latest.
ASMGMTC	The ASMGMTC parameter is in the //SYSIN part of the job. This parameter specifies SMS management class for allocation of the Automation Server Control file. This class name is optional, but the parameter must be specified. Specify a null ('') if it is not used. The ASMGMTC and ASSTORC parameters are used to specify where the data sets are to be allocated. If SMS is not enabled or you wish to specify volumes to be used, specify nulls for both ASMGMTC and ASSTORC and specify the volume names in ASVOLS.
ASSTORC	The ASSTORC parameter is in the //SYSIN part of the job. This parameter specifies SMS storage class for allocation of the Automation Server Control file. This class name is optional, but the parameter must be specified. Specify a null ('') if it is not used. See the ASMGMTC entry for details.
ASVOLS	The ASVOLS parameter is in the //SYSIN part of the job. This parameter specifies DASD volumes for allocation of the Automation Server Control file. The volumes specification is optional, but the parameter must be specified. Specify a null ('') if it is not used. The format of data in this parameter is '(volume)[,(volume)]...'. This parameter is placed in the DFSMSDSS RESTORE command OUTDYNAM() parameter. See the ASMGMTC entry for details. You must specify either ASMGMTC or ASSTORC, or ASVOLS.
GKBSCHMA	The GKBSCHMA parameter is in the //SYSIN part of the job. This parameter specifies the DB2 Schema name for the tables holding the Global Knowledge Base. The schema names must be less than or equal to seven characters in length. The first character must be an alphabetic character.
GKBUSCHM	The GKBUSCHM parameter is in the //SYSIN part of the job. This parameter specifies the DB2 Schema name for the tables holding the z/OS UNIX Global Knowledge Base. The schema names must be less than or equal to seven characters in length. The first character must be an alphabetic character.

LKBSCHMA	The LKBSCHMA parameter is in the //SYSIN part of the job. This parameter specifies the DB2 Schema name for the tables holding the Local Knowledge Base. The schema names must be less than or equal to seven characters in length. The first character must be an alphabetic character.
LKBUSCHM	The LKBUSCHM parameter is in the //SYSIN part of the job. This parameter specifies the DB2 Schema name for the tables holding the z/OS UNIX Local Knowledge Base. The schema names must be less than or equal to seven characters in length. The first character must be an alphabetic character.
REPSCHMA	The REPSCHMA parameter is in the //SYSIN part of the job. This parameter specifies the DB2 Schema name for the tables holding the Repository. The schema names must be less than or equal to seven characters in length. The first character must be an alphabetic character.
FLSCHMA	The FLSCHMA parameter is in the //SYSIN part of the job. This parameter specifies the DB2 Schema name for the tables holding the Inquisitor Import Filter data. The schema names must be less than or equal to seven characters in length. The first character must be an alphabetic character.

IQSCHEMAS	The IQSCHEMAS parameter is in the //SYSIN part of the job.
	
	You can specify multiple IQSCHEMAS values for this parameter. For example, you can set IQSCHEMAS = ’TFQ1 TFQ2’ and the customization job creates members DITFQ1, DITFQ2, and DXTFQ1, DXTFQ2. Besides using the IQSCHEMAS values for creating schema names, they are also used for creating table spaces and tables in DB2. For example, using value ’TFQ1’ results in creating these DB2 objects. The example shown here is for one table space and one table only:
	`tablespace WSTFQ1`
	`tablespace WSUTFQ1 (for z/OS UNIX)`
	`table TFQ1.TSYSTEM`
	`table UTFQ1.TSYSTEM (for z/OS UNIX)`
	
	The schema names must be less than or equal to five characters in length. The first character must be an alphabetic character.
	
	This parameter depends on the value specified in the MODE parameter. If IQSCHEMAS span multiple lines then use MODE=IQ for subsequent lines and then run HSISCUST for each IQSCHEMAS list of definitions. Here is a suggested approach:
	1. `PARM='ISPSTART CMD(%HSIICUST MODE=ALL)'`
	Include list of IQSCHEMAS. Run HSISCUST will create all jobs.
	2. `PARM='ISPSTART CMD(%HSIICUST MODE=IQ)'`
	Include new list of IQSCHEMAS. Run HSISCUST will create jobs for the new members in IQSCHEMAS. Members prefixed with ’HSIS’ and IQSCHEMAS jobs from (1) are not replaced. Repeat this step for additional list of IQSCHEMAS.

IQWMPRI	The IQWMPRI parameter is in the //SYSIN part of the job. This parameter specifies the primary allocation amount (in kilobytes) for the Inquisitor TMODULE table.
IQWLPRI	The IQWLPRI parameter is in the //SYSIN part of the job. This parameter specifies the primary allocation amount (in kilobytes) for the Inquisitor TLIBRARY table.
IQWMUPRI	The IQWMUPRI parameter is in the //SYSIN part of the job. This parameter specifies the primary allocation amount (in kilobytes) for the Inquisitor TMODULE table for z/OS UNIX.
IQWLUPRI	The IQWLUPRI parameter is in the //SYSIN part of the job. This parameter specifies the primary allocation amount (in kilobytes) for the Inquisitor TLIBRARY table for z/OS UNIX.
LOGGED	The LOGGED parameter is in the //SYSIN part of the job. This parameter determines whether data changes to IQSchema tables are recorded in the DB2 log. For DB2 Version 8, it must be set to ″Y″. For DB2 Version 9, it is recommended to set the parameter to ″N″.
BGKBTS, BGKBTS1, BGKBIX, BGKBIX1A, BGKBIX1B, BGKBIX1C, BGKUTS, BGKUIX, BLKBTS, BLKBIX, BLKUTS BREPTS, BREPTS1, BREPTS2, BREPIX, BREPIX1A, BREPIX1B, BREPIX1C, BREPIX1D, BREPIX2A, BREPIX2B, BREPIX2C, BREPIX2D, BREPIX2E, BREPIX2F, BREPIX2G BIQFTS, BIQFIX, BIQTS, BIQTS1 BIQIX, BIQIX1A, BIQIX1B, BIQIX1C, BIQIX1D, BIQIX1E, BIQIX1F, BIQIX1G, BIQIX1H, BIQIX1I, BIQIX1J, BIQIX1K, BIQUTS, BIQUIX	These buffer pool parameters are in the //SYSIN part of the job. These parameters specify the buffer pool definitions for the table spaces and indexes. For specific definitions of each buffer pool parameter, refer to the HSISCUST in the ’hlq’.SHSISAMP data set.

Once you have made the appropriate changes to the copy of the customization job, submit it to run. This job must create or reuse two output PDSE libraries and one sequential data set. These are:

JCLLIB
: The library contains JCL jobs to create DB2 resources, and to run the product’s functions.

PARMLIB
The library is referenced by jobs in the JCLLIB, and contains pre-tailored parameters.

UM.HLQIDS
This sequential data set, when first created, contains a single record. The data set is then populated by the Load to Repository job, and the contents, consisting of HLQ listing, is referenced by the Usage Monitor.

Use the JCLLIB members in this table to submit jobs to implement the product:

Table 1. JCLLIB members

HSISDB00	JCL SET statements to set symbols for data set high level qualifiers. This is included by other jobs.

Table 2. Post-installation jobs

HSISDB01	Job to define DB2 storage groups and the database name.
HSISDB02	Job to create the DB2 table spaces for the Global Knowledge Base.
HSISDB03	Job to load the Global Knowledge Base.
HSISDB04	Job to create the DB2 table spaces for the z/OS UNIX Global Knowledge Base.
HSISDB05	Job to load the z/OS UNIX Global Knowledge Base.
HSISDB06	Job to create the DB2 resources for the Local Knowledge Base and z/OS UNIX Local Knowledge Base.
HSISDB07	Job to create the DB2 resources for the Repository.
HSISDB11	Job to create the DB2 resources for the Inquisitor filters and to load the filters.
DI*	Jobs (per IQ schema) to create DB2 resources for the Inquisitor Import.
DX*	Jobs (per IQ schema) to create DB2 resources for the z/OS UNIX Inquisitor Import.
HSISGRNT	Job to grant general (non IQ schema) access to DB2 resources.
GI*	Jobs (per IQ schema) to grant access to DB2 resources.
GX*	Jobs (per IQ schema) to grant access to DB2 resources for z/OS UNIX.

Table 3. Capture phase

ZIQ*	Jobs (per IQ schema) to run the Inquisitor.
UIQ*	Jobs (per IQ schema) to run the z/OS UNIX Inquisitor.
HSISUMON	Job to run the Usage Monitor.

Table 4. Computation phase

ZIM*	Jobs (per IQ schema) to run the Inquisitor Import.
UIM*	Jobs (per IQ schema) to run the z/OS UNIX Inquisitor Import.
ZME*	Jobs (per IQ schema) to run the Match Engine.
UME*	Jobs (per IQ schema) to run the z/OS UNIX Match Engine.
ZLR*	Jobs (per IQ schema) to run the Load to Repository.
ULR*	Jobs (per IQ schema) to run the z/OS UNIX Load to Repository.

Table 4. Computation phase (continued)

HSISUIMP	Job to run the Usage Import.

Table 5. Reporting phase

HSISTCRR	Job to list the Tivoli Common Reporting Data Source and JDBC URL.
HSISBATR	Job to produce ad hoc reports.

Table 6. Housekeeping/utilities

HSIASALC	Job to allocate the Automation Server control file.
HSIASSCT	Job to run the Automation Server Scout utility.
HSISCOMB	Job to run combination of Inquisitor Import, Match Engine, and Load to Repository. This job is to be used as a sample within the Automation Server, and is found in PARMLIB (not JCLLIB).
HSISPTAG	Job to run the Product Tagging utility.
HSISREGN	Job to run the Region Association utility.
HSISINVA	Job to run the Add Inventory Association.
HSISASSD	Job to run the Delete Inventory Association.
HSISUSUM	Job to run the Usage Summary.
HSISUDEL	Job to run the Usage Deletion.
HSISDINV	Job to delete an inventory from the Repository.
HSISLLST	Job to create an HLQ listing for the Usage Monitor.
HSISMERG	Job to merge Repositories.
HSISTPRM	Job to update the Repository TPARAM table.
HSISMCMP	Job to compile Japanese messages for MMS.
HSISSCRT	Job to read SCRT CSV files and populate the Repository.
HSISKBT	Job to run the XML Export utility. **Note:** The XML output is for Tivoli Asset Management for IT.
HSISZCAT	Job to concatenate Usage Monitor data sets.

Table 7. Deployment for large sites

HSISUNLD	Job to unload the Repository tables.
HSISLOAD	Job to load the Repository tables .
HSISAM01	Job to create Asset mirror tables and views.
HSISAM02	Job to load AM02 Asset mirror tables.
HSISAM03	Job to load AM03 Asset mirror tables.
HSISAM04	Job to load AM04 Asset mirror tables.

Running post-installation jobs

If you are a new user, start by submitting job HSISDB01 from the JCLLIB. These jobs must be run on the system that contains the DB2 subsystem the database is to be defined on. Once that job has completed, continue submitting and running jobs

HSISDB02 through to HSISDB07, HSISDB11, DI*, DX*, HSISGRNT, GI* and GX*. Submit the next job after successful completion of the previous job. Completion codes of 4 are acceptable in these jobs.

After the completion of the customization jobs, the product must be made available to all systems that the various phases are run on.

Note: The APF authorization of the SHSIMOD1 library must be done on all systems that the product is to be run on

If changes are made to the JCLLIB and PARMLIB data sets, either manually or by rerunning the HSISCUST customization job, they must be redistributed to all systems that run the product.

Customizing tasks for Japanese messages

MVS Message Service (MMS) message information is provided for the Japanese language.

MMS message information for each language is contained in the following PDS:
- hlq.SHSIMJPN (Japanese)

The MMS message file must be compiled using the MMS message compiler. See the *MVS Assembler Services Guide* for more information on loading MMS messages.

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to compile MMS files. Member HSISMCMP contains JCL to run the job.

You must install the latest system runtime message file.

Here is an example of how to create a new Japanese system runtime message file and install it:
1. Compile MMS files into system runtime message file hlq.MMSJPN99
2. Create a new entry in PARMLIB called MMSLSTJ9
   ```
   DEFAULTS LANGCODE(JPN)
   LANGUAGE LANGCODE(JPN) DSN(SYS2.MMSJPN99) CONFIG(CNLJPN00)
   ```
3. Install the latest system runtime message file using this MVS system command
   ```
   SET MMS=J9
   ```

Note:

The components of the product listed here do not use the MVS message service MMS:
- Inquisitor Import
- Match Engine
- Load to Repository
- Usage Import
- Usage Summary
- Usage Deletion
- Delete Inventory
- Delete Repository

Japanese customers can enable Japanese messages to the MSGFILE for these components if the Language Environment NATLANG(JPN) option is in effect. For more information on the Language Environment MSGFILE and NATLANG options, refer to the *Language Environment Programming Reference (SA22–7562).* For more information on specifying Language Environment runtime options, refer to the *Language Environment Programming Guide (SA22–7561)*. For more information on setting NATLANG(JPN) as an installation default, refer to the *Language Environment Customization (SA22-7564).* When the NATLANG(JPN) option is not in effect, the components listed here always output messages in mixed case English.

For example, the Inquisitor Import JCL can be modified to produce Japanese messages (assuming the default installation NATLANG is not already set to Japanese) by modifying the following line in members beginning with ZIM (UIM for z/OS UNIX) and suffixed with the schema name :

```
//IMPORT  EXEC PGM=HSICIQIM,REGION=0M,TIME=1440
```

to:

```
//IMPORT  EXEC PGM=HSICIQIM,REGION=0M,TIME=1440,PARM='NATLANG(JPN)/'
```

Chapter 6. Post-installation tasks for Tivoli Common Reporting

These tasks need to be performed post-installation in order to use the reporting component in Tivoli Common Reporting.

Installing the DB2 JDBC driver and license JARs

See "Installing DB2 Connect (if required)" on page 18 on how to install DB2 Connect.

Before installing the JDBC JARs, ensure the Tivoli Integrated Portal (TIP) service is stopped.

To stop the server, left mouse click **Start** → **All Programs** → **Tivoli Common Reporting** → Stop **Tivoli Common Reporting** server.

To install the DB2 JDBC Driver and License JARs, copy the required JARs from the source directory to the destination directory. For the Linux on IBM System z platform, the JDBC JAR files must be uploaded in binary mode.

DB2 JARs required are:

db2jcc.jar
The DB2 Universal JDBC Driver.

db2jcc_license_cisuz.jar
The license file, `db2jcc_license_cisuz.jar`, contains licenses for Linux, Unix, Windows, IBM System i®, and IBM System z. The IBM System z license is required in order to establish a JDBC connection to DB2 running on z/OS.

The 'Personal Edition' installations of DB2 Connect do not contain the license `db2jcc_license_cisuz.jar`. They contain the alternate license `db2jcc_license_cu.jar`. This alternate license is not sufficient to connect to DB2 running on z/OS. To avoid this issue, ensure you install the Enterprise or Server editions of DB2 Connect that match the DB2 z/OS versions.

Source directory:

```
C:\Program Files\IBM\SQLLIB\java
```

Destination directory (Windows):

```
C:\IBM\tivoli\tip\products\tcr\lib\birts-runtime-
2_2_1\ReportEngine\plugins\org.eclipse.birt.report.data.oda.jdbc_2.2.
r22x_v20070919\drivers
```

Destination directory (Linux):

```
/opt/IBM/tivoli/tip/products/tcr/lib/birt-runtime-
2_2_1/plugins/org.eclipse.birt.report.data.oda.jdbc_2.2.
1.r22x_v20070919/drivers
```

Starting Tivoli Common Reporting

From the Start menu:

1. Start the Tivoli Common Reporting server.
2. Start the Tivoli Common Reporting browser.
3. Login to the Tivoli Common Reporting browser using the *user ID* and password configured when Tivoli Common Reporting was installed.

The above applies only to the Windows environment. For other platforms see the *IBM Tivoli Common Reporting User's Guide*

Transferring the Tivoli Common Reporting report package from the host system

Connect to the host system and copy the Tivoli Common Reporting report package.

1. C:\>ftp *host name* (Using command line FTP connect to the host system).
2. When prompted, enter your user name and password.
3. ftp > binary (Switch the FTP mode to binary).
4. ftp > get *'hlq.SHSITCR1(HSITCR)' HSITCR.zip*

   (Download the reports member).
5. ftp > exit.

   (Close the FTP connection).

Importing the Tivoli Common Reporting report package

The HSITCR.zip report package contains the design definitions and associated resource files for Tivoli Asset Discovery for z/OS reports. These can be imported using the Tivoli Common Reporting browser.

1. To view the reporting screen, expand **Reporting** and select **Common Reporting**.
2. Right mouse click on **Report Sets** and select **Import Report Package** from the dialog box.
3. Set Local file to the zip file downloaded in the previous step
4. Click on **Advanced Options** and set these options: **Overwrite** → **checked** and **Security Set** → **Tivoli Asset Discovery for z/OS.**
5. Click **Import**

The Tivoli Asset Discovery for z/OS reports will now appear in the report tree.

Determining Tivoli Common Reporting JDBC data sources

During the customizing of Tivoli Common Reporting, JDBC URLs must be provided to the Report Data Source Dialog.

The HSISCUST post-installation customization job creates JCL jobs in JCLLIB to list the JDBC URLs. Member HSISTCRR contains JCL to run the job. The output you can expect is shown here:

```
DATA_SOURCE  JDBC_URL___________________________________________________________
HSIzREP      jdbc:db2://exampsys.ibm.com:5000/QXPOMU1DE81:currentSchema=S17;
```

where the JDBC URL for DB2 has the following format:

```
jdbc:db2://<DB2 hostname>:<DB2 Port>/<DB location>:currentSchema=<schema name>;
```

This sample contains the JDBC URL for a Repository schema. Tivoli Common Reporting is configured to connect to this schema in the next step, "Configuring the Tivoli Common Reporting report data sources for JDBC"

Note: This example is for a hostname where the DB2 subsystem is located. If your site uses a different network setup, then the hostname that is listed may not be the correct one to use.

Configuring the Tivoli Common Reporting report data sources for JDBC

The following steps describe how to configure a JDBC data source for direct access. To configure to use JNDI see the Tivoli Common Reporting User's Guide.

1. Click on **Tivoli Common Reporting** window.
2. Click on **Discovery Administrator Reports**.
3. In the Reports window to the right of the menu, right click on the **Installation Verification Report** and select **Data Sources**.
4. Select the HSIzREP report data source from the Report Data Source dialog and click **Edit**.
5. Enter the required data:
   - Display Name - leave blank or use the default name HSIzREP
   - JDBC Driver - leave as DB2 driver: com.ibm.db2.jcc.DB2Driver
   - User ID (TSO user ID)
   - JDBC URL

     (only `jdbc:db2://exampsys.ibm.com:5000/QXPOMU1DE81:currentSchema=SI7;` is required)
   - Password (TSO password)
   - JNDI Name - leave blank
6. Click on **Save** to keep your data source settings.
7. Click on **Cancel** to exit the Report Data Sources dialog.

When a user is running their own copy of Tivoli Common Reporting, they can use their own TSO user ID and password. When running Tivoli Common Reporting in a server environment, it is recommended to create a separate TSO user ID specifically for Tivoli Common Reporting. This TSO user ID has its own password and is maintained by the security software on z/OS.

Verifying your installation

1. In the report navigation window, go to the navigation tab.
2. Navigate to the **Installation Verification Report** → **Report Sets** → **Tivoli Products** → **Tivoli Asset Discovery for z/OS** → **Discovery Administrator Reports.**
3. Click the icon just to the left of the Installation Verification Report.
4. Another browser window opens containing the Installation Verification Report.

When Tivoli Common Reporting has been successfully configured, the new browser window will display a report showing a list of schema names recognized by Tivoli Asset Discovery for z/OS by database type. It also verifies that the current schema setting in the JDBC URL is set to a Tivoli Asset Discovery for z/OS Repository.

If the browser window displays an error message, click on the link to see the report with errors. The error will be described at the bottom of the report. For example:

- "Cannot open the connection for the driver: `org.eclipse.birt.report.data.oda.jdbc`"

  For this error message, review the contents of the Report Data Source dialog set up in "Configuring the Tivoli Common Reporting report data sources for JDBC" on page 31
- "Cannot load JDBC Driver class: `com.ibm.db2.jcc.DB2Driver`"

  For this error message, check that the DB2 JDBC driver and license JARs have been copied correctly into the Tivoli Common Reporting driver directory.

After making the corrections, stop and start the Tivoli Reporting Server, and rerun the Installation Verification report.

Tivoli Common Reporting user roles

Tivoli Common Reporting can be configured to provide access to different combinations of reports and functions to each user. During Tivoli Common Reporting installation an administrator role is created by default. Logging in with this role allows you to access the user and group administration and report set authorizations features.

Two groups need to be created for Tivoli Asset Discovery for z/OS users to select the reports relevant to their job role.

Repository View Group: These reports are concerned with the software products discovered, where they are located, and who has been using them.

Administration View Group: These reports show the information relating to the Tivoli Asset Discovery for z/OS components. The data source verification. The data captured by the Inquisitor. The Knowledge Bases used by the Match Engine, and user software that is yet to be fully identified. This view also has access to all repository view reports.

For the repository view add a group called TADzREP using the Manage Group function. For the Administration view add a group called TADzADMIN. Then using the Administrative Group Roles function, add the two groups and provide each of them with the *tcrOperator role*. This allows members of these groups to log on to Tivoli Common Reporting. Add Users using the Manage Users function and then add them to their relevant job role group.

At this stage new users can login to Tivoli Common Reporting, but will not see any reports when they select Common Reporting. The groups need to be authorized to access the nodes in the report hierarchy. Right mouse click on each node in the report hierarchy and select Authorizations from the popup menu. Add the group to the authorized user and groups list. When a group is added to the authorization list it has a default role of *tcrReportViewer*. Select each added group and set the roles as described in the table below.

Table 8. TADzREP

Report Navigation Hierarchy	Roles
Report Sets	tcrSetViewer 1
Tivoli Products	tcrSetViewer
Tivoli Asset Discovery for z/OS	tcrSetViewer
Advanced Reports	tcrSnapshotAuthor, tcrReportDistributor 2
Asset Reports	tcrSnapshotAuthor, tcrReportDistributor
Standard reports	tcrSnapshotAuthor, tcrReportDistributor
Tivoli Common Reporting	tcrReportViewer 3

Table 9. TADzADMIN

Report Navigation Hierarchy	Roles
Report Sets	tcrSetViewer 1
Tivoli Products	tcrSetViewer
Tivoli Common Reporting	tcrReportViewer 3

1 Allows the user to navigate to a report set. Using this role allows a user access to a child report set without granting full access to the parent sets.

2 Allows the user to view, but not modify the reports, create and distribute snapshots and schedule a report.

3 Allows the user to view, but not modify the report set.

For further details on Users and Groups see "Administering users" in the *IBM Tivoli Common Reporting User's Guide*.

Chapter 7. Migration

If your site is currently running Version 7 Release 1 Limited Availability Fix, Dorana Version 5 Release 4 or IBM Tivoli License Compliance Manager for z/OS, and if you want to migrate your application data to Tivoli Asset Discovery for z/OS Version 7.2, then you should follow the instructions as listed here.

IBM Tivoli License Compliance Manager for z/OS conversion

The data conversion process converts MONDETL and SURVDATA files from Tivoli License Compliance Manager for z/OS; Version 3 Release 2, Version 4 Release 1, and Version 4 Release 2, to the format required by this product. No other data is converted.

The conversion process allows you to control when each system within the enterprise is to be converted from Tivoli License Compliance Manager for z/OS to this product. This staged approach to conversion allows both products to run concurrently until you decide to cut over to Tivoli Asset Discovery for z/OS. This product and DB2 can be installed on the system of your choice, with Tivoli License Compliance Manager for z/OS continuing to run on all existing systems within the enterprise.

Conversion allows you to start using the new product database and reporting capabilities against your entire enterprise without having to uninstall Tivoli License Compliance Manager for z/OS and install this product on every system. It enables the data generated by your existing Tivoli License Compliance Manager for z/OS data gatherers to be used by the new product. Once you are satisfied with the converted data, you can discontinue the Tivoli License Compliance Manager for z/OS data analysis and reporting process that runs on each system where it is installed.

The record format of the SURVDATA file has changed between Version 3 Release 2, and Version 4 Release 1. The Surveyor conversion process only converts Tivoli License Compliance Manager for z/OS Version 4 Release 1 format SURVDATA files into Tivoli Asset Discovery for z/OS Version 7 Release 2 format.

For Tivoli License Compliance Manager for z/OS SURVDATA Version 3 Release 2 files, an intermediate step called the UPGRADER is required and is already included in the conversion job. The UPGRADER converts Tivoli License Compliance Manager for z/OS SURVDATA Version 3 Release 2 format files into Tivoli License Compliance Manager for z/OS SURVDATA Version 4 Release 1 format.

By default, the SURVOUT DD is a temporary PDS/E data set. This is adequate for most conversions. For a larger Surveyor file that needs to be converted by the UPGRADER, increase the size of the SURVOUT data set by approximately 40% of the SURVIN data set.

Releases prior to Version 3 Release 2 cannot be upgraded, and therefore cannot be converted.

Before you use the product for the first time, you need to step through the section "Creating post-installation jobs" on page 18. Once you have made the appropriate changes to HSISCUST, submit this job to run.

The job must create or reuse two output PDSE libraries and one sequential data set. These are:

1. JCLLIB - The library contains JCL jobs to create DB2 resources, and to run the product's functions.
2. PARMLIB - The library is referenced by jobs in the JCLLIB, and contains pre-tailored parameters.
3. UM.HLQIDS - This sequential data set, when first created, contains a single record. The data set is then populated by the Load to Repository job, and the contents, consisting of HLQ listing, are referenced by the Usage Monitor.

For the conversion implementation, use the JCLLIB members to submit jobs in the sequence as listed here:

HSISDB01	Job to define DB2 Storage Groups and the Database name.
HSISDB02	Job to create the DB2 table spaces for the Global Knowledge Base.
HSISDB03	Job to load the Global Knowledge Base.
HSISDB04	Job to create the DB2 table spaces for the z/OS UNIX Global Knowledge Base.
HSISDB05	Job to load the z/OS UNIX Global Knowledge Base.
HSISDB06	Job to create the DB2 resources for the Local Knowledge Base and z/OS UNIX Local Knowledge Base.
HSISDB07	Job to create the DB2 resources for the Repository.
HSISDB11	Job to create the DB2 resources for the Inquisitor filters and to load the filters.
DI*	Jobs (per IQ schema) to create DB2 resources for the Inquisitor Import.
DX*	Jobs (per IQ schema) to create DB2 resources for the z/OS UNIX Inquisitor Import.
HSISGRNT	Job to grant general (non IQ schema) access to DB2 resources.
GI*	Job (per IQ schema) to grant access to DB2 resources.
GX*	Job (per IQ schema) to grant access to DB2 resources for z/OS UNIX.

Running the conversion jobs:

HSISM2D	Monitor conversion. Run time for this job depends on the number of records to be converted. It is recommended to run this job during off-peak periods.
HSISS2D1	Surveyor conversion for Tivoli License Compliance Manager for z/OS Version 4 Release 1 or Version 4 Release 2. Run time for this job depends on the number of records to be converted. It is recommended to run this job during off-peak periods.

HSISS2D2	Surveyor conversion for Tivoli License Compliance Manager for z/OS Version 3 Release 2. Note that this job contains an intermediate step called the UPGRADER. Run time for this job depends on the number of records to be converted. It is recommended to run this job during off-peak periods.

Loading the converted Tivoli License Compliance Manager for z/OS data to Tivoli Asset Discovery for z/OS:

ZIM*	Job (per IQ schema) to run the Inquisitor Import. Input data is from output of HSISS2D1 or HSISS2D2
HSISUIMP	Job to run the Usage Import. Input data is from output of HSISM2D

Post-conversion tasks

- You can continue to run your existing Monitor and Surveyor as before.
- Convert and import Tivoli License Compliance Manager for z/OS Monitor and Surveyor data into the new Tivoli Asset Discovery for z/OS Version 7 Release 2.
- When Tivoli Asset Discovery for z/OS Version 7 Release 2 has been fully implemented, and the new Usage Monitor and Inquisitor Scan are ready for use, the Tivoli License Compliance Manager for z/OS version of the Monitor and Surveyor can be discontinued.
- Once you have discontinued using the previous product, you can proceed to run the remaining jobs as described in the Capture, Computation, and Reporting phases.

Tivoli License Compliance Manager for z/OS Exporter compatibility

To provide similar functionality as the Exporter, four members are provided in SHSIEXEC data set. These members query information stored in DB2, and return output stored in the four Tivoli License Compliance Manager for z/OS Exporter DSECT formats.

Table 10. SHSIEXEC queries

REXX samples in SHSIEXEC	Exporter Tivoli License Compliance Manager for z/OS DSECT	Description
HSIIXPMO	XPMODS	Installed Load Modules
HSIIXPPR	XPPRODS	Installed Products
HSIIXPUM	XPUSAGEM	Load Module use
HSIIXPUP	XPUSAGEP	Product use

Some fields cannot be generated as they were in Tivoli License Compliance Manager for z/OS. These fields are marked with a ? (question mark) to signify this.

The description, function, and JCL required to run these samples is documented in each member.

The REXX samples TRNMODS, TRNUSGM, and TRNUSGP are provided with Tivoli License Compliance Manager for z/OS. They transform each record into transaction file format for import into a spreadsheet. If you run these during migration, the date fields not available from Tivoli Asset Discovery for z/OS are displayed as ’6F’; the hexadecimal EBCDIC representation of ’?’.

Before using these samples, ensure the DB2 REXX environment for the target DB2 system has been successfully set up .

Migration from Dorana Version 5 Release 4

Pre migration recommendations

For improved migration performance, it is recommended that the tasks listed here are run on the existing Dorana database.

- Run the Usage Summary job.
- Run the Usage Deletion job. It is recommended that you delete any Usage data older than 12 months.

  Refer to Appendix C, “Deletion of obsolete libraries in Dorana Version 5 Release 4 Repository,” on page 205 in Appendix C for information on how to delete obsolete libraries prefixed with ″!″ that are found in the Repository tables.
- Reorganize the Repository table spaces.

Migration tasks

In order to migrate Dorana Version 5 Release 4 to Tivoli Asset Discovery for z/OS Version 7 Release 2, complete the step as explained in section “Creating post-installation jobs” on page 18. The HSISCUST parameters listed below must be customized with different values as used in your Dorana Version 5 Release 4 system. This is to avoid any duplicate names that have already been used.

1. DB - database
2. Schema Names

   ```
   GKBSCHMA
   GKBUSCHM
   LKBSCHMA
   LKBUSCHM
   REPSCHMA
   FLSCHMA
   ```

Note: For this migration.:

- Data is unloaded from Dorana.
- Data is loaded into Tivoli Asset Discovery for z/OS.
- New table spaces are created.
- New tables and indexes are created.
- Columns are added to existing tables.
- Column sizes are modified.

Once you have made the appropriate changes to HSISCUST, submit this job to run.

The job must create or reuse two output PDSE libraries and one sequential data set. These are:

1. JCLLIB - The library contains JCL jobs to create DB2 resources, and to run the product’s functions.

2. PARMLIB - The library is referenced by jobs in the JCLLIB, and contains pre-tailored parameters.
3. UM.HLQIDS - This sequential data set, when first created, contains a single record. The data set is then populated by the Load to Repository job, and the contents, consisting of HLQ listing, are referenced by the Usage Monitor.

For the migration implementation, use the JCLLIB members to submit jobs in the sequence as listed here:

HSISDB01	Job to define DB2 storage groups and the database name.
HSISDB02	Job to create the DB2 table spaces for the Global Knowledge Base.
HSISDB03	Job to load the Global Knowledge Base.
HSISDB04	Job to create the DB2 table spaces for the z/OS UNIX Global Knowledge Base.
HSISDB05	Job to load the z/OS UNIX Global Knowledge Base.
HSISDB06	Job to create the DB2 resources for the Local Knowledge Base and z/OS UNIX Local Knowledge Base.
HSISDB07	Job to create the DB2 resources for the Repository.
HSISDB11	Job to create the DB2 resources for the Inquisitor filters and to load the filters.
DI*	Jobs (per IQ schema) to create DB2 resources for the Inquisitor Import.
DX*	Jobs (per IQ schema) to create DB2 resources for the z/OS UNIX Inquisitor Import.
HSISGRNT	Job to grant general (non IQ schema) access to DB2 resources.
GI*	Job (per IQ schema) to grant access to DB2 resources.
GX*	Job (per IQ schema) to grant access to DB2 resources for z/OS UNIX.

If your site does not have any data from LKB and LKU to migrate, then ignore running the following 4 jobs (HSISMI20, HSISMI21,HSISMI22, and HSISMI23).

HSISMI20	LKB - Unload from Dorana
HSISMI21	LKB - Reload to Tivoli Asset Discovery for z/OS.
HSISMI22	LKU - Unload from Dorana
HSISMI23	LKU - Reload to Tivoli Asset Discovery for z/OS.

HSISMI24	Repository - unload 24 tables from Dorana. Run time for this job depends on the amount of data to be unloaded. It is recommended to run this job during off-peak periods.
HSISMI25	Repository - reload 24 tables to Tivoli Asset Discovery for z/OS. Run time for this job depends on the amount of data to be loaded. It is recommended to run this job during off-peak periods.
HSISMI30	Populate the newly defined tables in the Repository for the Tivoli Asset Discovery for z/OS. Run time for this job is dependent on implementing the tasks in the pre migration recommendations section. It is recommended to run this job during off-peak periods.

Post-migration tasks

- You need to run the Usage Import job (HSISUIMP), or the SCRT import utility job (HSISSCRT). This allows the Repository Asset tables to be populated correctly and the Tivoli Common Reporting reports to be generated with the correct information.
- You can continue to run your existing Usage Monitor and Inquisitor Scan as before.
- Importing Dorana Version 5 Release 4 Usage and Inquisitor data into the new Tivoli Asset Discovery for z/OS Version 7 Release 2 is tolerated, as the data structures are compatible.
- When Tivoli Asset Discovery for z/OS Version 7 Release 2 has been fully implemented, and the new Usage Monitor and Inquisitor Scan are ready for use, the previous version of the Usage Monitor and Inquisitor Import can be discontinued.
- Once you have discontinued using the previous product, you can proceed to run the remaining jobs as described in the Capture, Computation, and Reporting phases.

  DB2 table spaces based on schemas used in Dorana may now be dropped.
- Reorganization and backup of the new database is also recommended.

Migration from Version 7 Release 1 Limited Availability Fix

Pre-migration recommendations

For improved migration performance, it is recommended that the tasks listed here are run on the existing database, Version 7 Release 1 Limited Availability Fix:

- Run the Usage Summary job
- Run the Usage Deletion job
- Reorganize the Repository table spaces

Migration tasks

In order to migrate Version 7 Release 1 Limited Availability Fix to Tivoli Asset Discovery for z/OS Version 7 Release 2, complete the step as explained in section "Creating post-installation jobs" on page 18. The HSISCUST parameters listed here must be customized with identical values as used in your Version 7 Release 1 Limited Availability Fix system:

1. DB - database
2. Schema Names

   ```
   GKBSCHMA
   GKBUSCHM
   LKBSCHMA
   LKBUSCHM
   REPSCHMA
   FLSCHMA
   ```

Note: For this migration:

- No application data is migrated from the previous release.
- A new table space is created.
- New tables and indexes are created.
- The index is modified.
- Columns are added to existing tables.

- Column sizes are modified.

Once you have made the appropriate changes to HSISCUST, submit this job to run.

The job must create or reuse two output libraries and one sequential data set. These are:

1. JCLLIB - The library contains JCL jobs to create DB2 resources, and to run the product's functions.
2. PARMLIB - The library is referenced by jobs in the JCLLIB, and contains pre-tailored parameters.
3. UM.HLQIDS - This sequential data set, when first created, contains a single record. The data set is then populated by the Load to Repository job, and the contents, consisting of HLQ listing, are referenced by the Usage Monitor.

For the migration implementation, use the JCLLIB members to submit jobs in the sequence as listed here:

HSISMI10	Create/Alter DB2 resources: a) Add new column in Local Knowledge Base and Local Knowledge Base for z/OS UNIX. b) Add three new indexes for table TSCORPAT in the Local Knowledge Base. c) Expand sizes of existing column in Local Knowledge Base and Local Knowledge Base for z/OS UNIX. d) For the Repository: • Create new table space VAGGR. • Create 11 new tables and associated indexes. • Add new columns to existing tables. • Add new column to an existing index. • Expand size of an existing column. **Note:** Table spaces VSHARE, VUSEMTD, WLOCLKB5. WLOCLKBU, and indexes REPschema.IUSEMTD1 and REPschema.IVENDOR1, are set to advisory REORG-pending status (AREO*).
HSISMI11	Drop DB2 Resources in GKB, GKU, IQF, ME: a) Drop table spaces for Global Knowledge Base. b) Drop table spaces for Global Knowledge Base for z/OS UNIX. c) Drop table spaces for Inquisitor filters. d) Drop table spaces for Match Engine.
HSISDB02	Job to create the DB2 table spaces for the Global Knowledge Base.
HSISDB03	Job to load the Global Knowledge Base.
HSISDB04	Job to create the DB2 table spaces for the z/OS UNIX Global Knowledge Base.
HSISDB05	Job to load the z/OS UNIX Global Knowledge Base.
HSISDB11	Job to create the DB2 resources for the Inquisitor filters and to load the filters.

HSISMI30	Populate the newly defined tables in the Repository for the Tivoli Asset Discovery for z/OS. Run time for this job is dependent on implementing the tasks in the pre migration recommendations section. It is recommended to run this job during off-peak periods.
HSISGRNT	Job to grant general (non IQ schema) access to DB2 resources.
GI*	Job (per IQ schema) to grant access to DB2 resources.
GX*	Job (per IQ schema) to grant access to DB2 resources for z/OS UNIX.

Note:

1. All table spaces and indexes in Version 7 Release 2 are set to have values for the secondary quantity space allocation (SECQTY) of ″-1″. You can set all table spaces and indexes to have this value after completing the migration by using the DB2 command ′ALTER TABLESPACE′ and ′ALTER INDEX′. This task is optional and it is recommended that you consult your DB2 specialist.
2. All IQ table spaces in Version 7 Release 2 can be set to have values of ′NOT LOGGED′, and are only applicable to DB2 Version 9 (not supported in DB2 Version 8). You can set all table spaces to have this value after completing the migration by using the DB2 command ′ALTER TABLESPACE′. This task is optional and it is recommended that you consult your DB2 specialist.

Post-migration tasks

- You need to run the Usage Import job (HSISUIMP), or the SCRT import utility job (HSISSCRT). This allows the Repository Asset tables to be populated correctly and the Tivoli Common Reporting reports to be generated with the correct information.
- You can continue to run your existing Usage Monitor and Inquisitor Scan as before.
- Importing Version 7 Release 1 Limited Availability Fix Usage and Inquisitor data into Tivoli Asset Discovery for z/OS Version 7 Release 2 is tolerated, as the data structures are compatible.
- When Tivoli Asset Discovery for z/OS Version 7 Release 2 has been fully implemented, and the new Usage Monitor and Inquisitor Scan are ready for use, the previous version of the Usage Monitor and Inquisitor Import can be discontinued.
- Once you have discontinued the previous release, you can proceed to run the remaining jobs as described in the Capture, Computation, and Reporting phases.
- Some table spaces and indexes are set to advisory REORG-pending (AREO*) status as a result of the migration. To set the table spaces and indexes back to their normal (RW) state, a reorganization of these resources needs to be performed.
- Reorganization and backup of the new product is also recommended.

DB2 objects affected by migration

The table lists DB2 resources affected by application data migration. It is provided as a reference, and definitions for these DB2 objects can be found in the PARMLIB members.

Schema	Table Name	Dorana V5.4 T/S	V7.1 Limited Availability Fix T/S	V7.2 T/S
GKBSchema	TPRODUCT	WPDTGKB5	WPDTGKB5	WPDTGKB
GKBSchema	TPARAM	WRULGKB5	WRULGKB5	WRULGKB
GKBSchema	TPRSMAP	N/A	N/A	WRULGKB
GKBSchema	TPTFFMID	WRULGKB5	WRULGKB5	WRULGKB
GKBSchema	TRULES	WRULGKB5	WRULGKB5	WRULGKB
GKBSchema	TSCORPAT	WSCPGKB5	WSCPGKB5	WSCPGKB
GKBSchema	PRODUCT	N/A	N/A	WVDRGKB
GKBSchema	TPRODLINK	N/A	N/A	WVDRGKB
GKBSchema	TVENDOR	WVDRGKB5	WVDRGKB5	WVDRGKB
GKBSchema	TVERSION	WVERGKB5	WVERGKB5	WVERGKB
GKUSchema		WOPTGKBU	deprecated	deprecated
GKUSchema		WPTNGKBU	deprecated	deprecated
GKUSchema		WPVRGKBU	deprecated	deprecated
GKUSchema		WVPVGKBU	deprecated	deprecated
GKUSchema	TPRODUCT	WPDTGKBU	WPDTGKBU	WPDTGKU
GKUSchema	TPARAM	WRULGKBU	WRULGKBU	WRULGKU
GKUSchema	TPTFFMID	WRULGKBU	WRULGKBU	WRULGKU
GKUSchema	TRULES	WRULGKBU	WRULGKBU	WRULGKU
GKUSchema	TSCORPAT	WSCPGKBU	WSCPGKBU	WSCPGKU
GKUSchema	TVENDOR	WVDRGKBU	WVDRGKBU	WVDRGKU
GKUSchema	TVERSION	WVERGKBU	WVERGKBU	WVERGKU
LKBSchema	TPARAM	WLOCLKB5	WLOCLKB5	WLOCLKB
LKBSchema	TPRODUCT	WLOCLKB5	WLOCLKB5	WLOCLKB
LKBSchema	TPTFFMID	WLOCLKB5	WLOCLKB5	WLOCLKB
LKBSchema	TRULES	WLOCLKB5	WLOCLKB5	WLOCLKB
LKBSchema	TSCORPAT	WLOCLKB5	WLOCLKB5	WLOCLKB
LKBSchema	TVENDOR	WLOCLKB5	WLOCLKB5	WLOCLKB
LKBSchema	TVERSION	WLOCLKB5	WLOCLKB5	WLOCLKB
LKUSchema	TPARAM	WLOCLKBU	WLOCLKBU	WLOCLKU
LKUSchema	TPRODUCT	WLOCLKBU	WLOCLKBU	WLOCLKU
LKUSchema	TPTFFMID	WLOCLKBU	WLOCLKBU	WLOCLKU
LKUSchema	TRULES	WLOCLKBU	WLOCLKBU	WLOCLKU

Schema	Table Name	Dorana V5.4 T/S	V7.1 Limited Availability Fix T/S	V7.2 T/S
LKUSchema	TSCORPAT	WLOCLKBU	WLOCLKBU	WLOCLKU
LKUSchema	TVENDOR	WLOCLKBU	WLOCLKBU	WLOCLKU
LKUSchema	TVERSION	WLOCLKBU	WLOCLKBU	WLOCLKU
IQSchema	TCSECT	WC*	WC*	WC*
IQSchema	TDECISION	WD*	WD*	WD*
IQSchema	TLIBRARY	WL*	WL*	WL*
IQSchema	TMODULE	WM*	WM*	WM*
IQSchema	TMIGREPORT	WR*	WR*	WR*
IQSchema	TPARAM	WS*	WS*	WS*
IQSchema	TSYSTEM	WS*	WS*	WS*
IQUSchema		WCU*	deprecated	deprecated
IQUSchema	TDECISION	WDU*	WDU*	WDU*
IQUSchema	TLIBRARY	WLU*	WLU*	WLU*
IQUSchema	TMODULE	WMU*	WMU*	WMU*
IQUSchema	TMIGREPORT	WRU*	WRU*	WRU*
IQUSchema	TPARAM	WSU*	WSU*	WSU*
IQUSchema	TSYSTEM	WSU*	WSU*	WSU*
MESchema		WTOTALS	WTOTALS	deprecated
MESchema		WUNIDLM	WUNIDLM	deprecated
MESchema	TIDENTLMOD	WIDENTLM	WIDENTLM	deprecated
MESchema	TINTERLIBPRODUCTID	WINTLPDT	WINTLPDT	deprecated
MESchema	TINTERVERPRODUCTID	WINTVPDT	WINTVPDT	deprecated
MESchema	TXLIBSTOP	WXLIBSTP	WXLIBSTP	deprecated
IQFSchema	TCOMPILERS	WIQFILTR	WIQFILTR	WIQFILTR
IQFSchema	TIQFILTERS	WIQFILTR	WIQFILTR	WIQFILTR
IQFSchema	TPARAM	WIQFILTR	WIQFILTR	WIQFILTR
IQFSchema	TUXFILTERS	WIQFILTR	WIQFILTR	WIQFILTR
IQFSchema	TXPCMODULES	WIQFILTR	WIQFILTR	WIQFILTR
IQFSchema	TXPCSPEC	WIQFILTR	WIQFILTR	WIQFILTR
IQFSchema	TXVENDORS	WIQFILTR	WIQFILTR	WIQFILTR
REPSchema	NODE	N/A	N/A	VAGGR
REPSchema	NODE_CAPACITY	N/A	N/A	VAGGR
REPSchema	PRODUCT	N/A	N/A	VAGGR
REPSchema	PRODUCT_INSTALL	N/A	N/A	VAGGR

Schema	Table Name	Dorana V5.4 T/S	V7.1 Limited Availability Fix T/S	V7.2 T/S
REPSchema	PRODUCT_NODE_CAPACIT	N/A	N/A	VAGGR
REPSchema	PRODUCT_USE	N/A	N/A	VAGGR
REPSchema	SYSTEM	N/A	N/A	VAGGR
REPSchema	SYSTEM_NODE	N/A	N/A	VAGGR
REPSchema	TACCOUNT	N/A	N/A	VSHARE
REPSchema	TUSEPRS	N/A	N/A	VSHARE
REPSchema	PRODUCT_USE_DETAIL	N/A	N/A	VSHARE
REPSchema	TJOBDATA	VJOBDATA	VJOBDATA	VJOBDATA
REPSchema	TMODULE	VMODULE	VMODULE	VMODULE
REPSchema	TUSEMTD	VUSEMTD	VUSEMTD	VUSEMTD
REPSchema	TINVCTL	VINVCTL	VSHARE	VSHARE
REPSchema	TINVREG	VINVREG	VSHARE	VSHARE
REPSchema	TIQHISTORY	VIQHIST	VSHARE	VSHARE
REPSchema	TLIBRARY	VLIBRARY	VSHARE	VSHARE
REPSchema	TLPAR	VLPAR	VSHARE	VSHARE
REPSchema	TPARAM	VPARAM	VSHARE	VSHARE
REPSchema	TPERIODS	VPERIOD	VSHARE	VSHARE
REPSchema	TPOVINV	VPOVINV	VSHARE	VSHARE
REPSchema	TPOVLIB	VPOVLIB	VSHARE	VSHARE
REPSchema	TPRODUCT	VPRODUCT	VSHARE	VSHARE
REPSchema	TREGCLASS	VREGCLAS	VSHARE	VSHARE
REPSchema	TREGION	VREGION	VSHARE	VSHARE
REPSchema	TREGLEAF	VREGLEAF	VSHARE	VSHARE
REPSchema	TUIMPORTCTRL	VUIMPCTL	VSHARE	VSHARE
REPSchema	TUSELIB	VUSELIB	VSHARE	VSHARE
REPSchema	TUSEPO	VUSEPO	VSHARE	VSHARE
REPSchema	TUSEPOV	VUSEPOV	VSHARE	VSHARE
REPSchema	TUSEPOVLIB	VUPOVLIB	VSHARE	VSHARE
REPSchema	TUSERDATA	VUSRDATA	VSHARE	VSHARE
REPSchema	TVENDOR	VVENDOR	VSHARE	VSHARE
REPSchema	TVERSION	VVERSION	VSHARE	VSHARE

Chapter 8. Inquisitor for z/OS

The Inquisitor provides the ability to scan PDS and PDSE load libraries, and to extract information regarding the contents of those libraries. Data output by the Inquisitor becomes input to the Software Inventory creation.

While the simplest Inquisitor request is to scan all online load libraries, facilities are provided to limit processing to specific subsets of libraries.

Inquisitor JCL

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the z/OS Inquisitor. Members beginning with ZIQ, and suffixed with the Inquisitor schema name contain JCL to run the Inquisitor for a given schema.

Run time for this job depends on the number of volumes and libraries to be scanned. It is recommended to run this job during off-peak periods.

To allocate the MCDS, if any requests contain the REMIGRATE or NOML2 operands, supply an additional DD statement like this one:

```
//MCDS    DD  DSN=DFHSM.MCDS,DISP=SHR
```

Multiple MCDS files are now supported by using additional DD statements MCDS2, MCDS3, and MCDS4. The MCDS is the migration control data set of IBM's Hierarchical Storage Manager data management, DFHSM or DFSMShsm™, hereafter referred to as HSM.

To allocate the necessary files required to interface to FDRABR, if any requests contain the ABRMIG or ABRARC operands, supply these two additional DD statements:

```
//ABRPRINT DD  SYSOUT=*
//ABRIN    DD  UNIT=VIO,SPACE=(TRK,1)
```

Modify the JOB statement to suit the installation standards. Selection of a MSGCLASS which is held is useful.

Specifying a generous REGION in the job statement is recommended to remove region size as a possible cause of problems. MCDS processing uses about 700 kilobytes more region storage than might otherwise be used.

If you choose to run SCANPGM requests, ensure that the MEMLIMIT value is large enough to permit the relevant Program Binder processing of the largest program object that the Inquisitor encounters.

The optional program parameter allows the specification of a report message level, a job run identifier, and an override to the system identifier.

The program parameter details are:

- If more than one parameter setting is specified, the settings must be separated by a comma.

- DSNMSG requests that messages relating to processed data sets, which might otherwise be suppressed, are to be logged in the SYSPRINT report.
- PGMMSG requests that messages relating to processed programs, which might otherwise be suppressed, are to be logged in the SYSPRINT report.
- ALLMSG requests both DSNMSG and PGMMSG message logging.
- NOAPF specifies that the Inquisitor is to run in an environment which is not APF authorized.
- The SID value is up to four characters long, and specifies the system identifier to be contained in the Inquisitor's output data. If the SID identifier override is omitted, the system SMF identifier is used. The SID parameter setting is for use when the SMF system identifier of a system is not unique.

  Syntax:

  ```
  SID=SYS2
  ```
- The RUN value is up to eight characters long, and specifies an optional run identifier, which is contained in the output header record.

  Syntax:

  ```
  RUN=SPECIAL
  ```
- The PLX parameter is used to identify if the Inquisitor data being collected is a SYSPLEX. The value is either Y or N. If the PLX parameter is not used, the default value of N is created in the Inquisitor header record.

  Syntax:

  ```
  PLX=Y | N
  ```

Modify the high level qualifier of the STEPLIB DSNAME so that the correct load library is referenced. If FDRABR is used, ensure the load library contains the HSIPINQ and HSIPINQA programs.

Ensure the HSIPZIP output file has an appropriate data set name specified, and the UNIT and SPACE are correct.

The output volumes of an Inquisitor run are large, depending on the data present in the installation and the scope of specified requests.

It might be necessary to experiment with a limited scan before estimating the requirements of a complete scan.

Use control statement operands to limit the scope of Inquisitor requests, as appropriate.

Files used by the Inquisitor

SYSPRINT

The report file. The minimum LRECL is 81, plus 4 for the RDW, if variable length records are used. The maximum LRECL is 255. The flexible DCB support means the report can be easily redirected to a data set if the need arises.

HSIPZIP

The output file containing compressed Inquisitor data. It is written using a variable length record format. DCB information is to be provided in order to ensure optimal use of DASD space.

- For a 3390, the use of RECFM=VB,LRECL=27994,BLKSIZE=27998 is advised.

- For a 3380, the use of RECFM=VB,LRECL=23472,BLKSIZE=23476 is advised.

The HSIPZIP file must not undergo any translation when being transferred, whatever the architecture of the target system. Only BINARY transfers are to be used to transport the file.

HSIPOUT
The optional output file containing uncompressed Inquisitor data. It is not specified in the packaged sample, as the use of HSIPZIP is preferred, due to its reduced space requirements. HSIPOUT also contains variable length records. The program supplies the appropriate LRECL. By default, system determined block size is used.

If the Inquisitor output is to be directed to a compressible extended-format data set, then the HSIPOUT file must be used. The HSIPZIP file employs some update-in-place processing which prevents the use of DFSMS™ compression.

SYSIN
The request input file. Fixed length, variable length, and undefined record formats are processed. Short records are extended to 72 bytes in length with blanks, if necessary.

MCDS
This file allocates the HSM MCDS data set, and is required if any requests contain the REMIGRATE or NOML2 operands. Further, if supplied for other requests, it is used to avoid the recall of data sets which are not load libraries. If the HSM MCDS is spread over more than one data set, the DD names MCDS2, MCDS3, and MCDS4 are to be used consecutively to allocate all the MCDS data sets in key range order.

ABRIN
The SYSIN file of the FDRABRP utility program is required if any requests contain the ABRMIG or ABRARC operands. It is primed by the Inquisitor during execution. For this reason, a single track VIO file is an ideal allocation.

ABRPRINT
The SYSPRINT file of the FDRABRP utility program is required if any requests contain the ABRMIG or ABRARC operands. It is an output-only file, and is not processed by the Inquisitor.

Functions performed by the Inquisitor

The Inquisitor is a program, run in a batch job, which collects information about executable software existing in a z/OS DASD subsystem. The resulting file is used as input for data import post processing.

The Inquisitor can process many requests in a single program run. Output from all requests of a given run is contained in the same file. For example:

```
SCANDIR STOGROUP(*)
SCANDIR NONSMS
```

In an installation where some DASD volumes are SMS-managed, these two requests produce, in a single request, a file with the same size and data content as this example:

```
SCANDIR
```

These function request verbs are currently available:

SCANCMD
: The SCANCMD verb is used to allow command syntax and operand consistency to be checked by the Inquisitor without initiating Load Library processing. It performs a parse only operation, although output files are opened.

  Error messages relating to syntax and operand errors are produced as usual. This verb is of use while users are formulating the best request combination to implement for any given system.

SCANDIR
: The SCANDIR verb is used to collect data from program library directory entries. Contents of program members are not accessed.

  Compared to SCANPGM, its reduced data collection allows it to run faster. Although all syntactically correct operands are allowed, some operands relating to data from member contents are ignored during processing. SCANDIR collects all of the information needed for automated software identification, and is the verb of choice for a production environment.

SCANPGM
: The SCANPGM verb collects all data collected by SCANDIR, as well as information from member contents. Such information relates to program structure and history.

  Your IBM representative may request SCANPGM output data to assist with problem diagnosis and resolution.

SCANMOD
: The SCANMOD verb performs the same function as SCANPGM, except that PDSE libraries are excluded from processing. Only load modules in the PDS libraries are processed.

SCANOBJ
: The SCANOBJ verb performs the same function as SCANPGM, except that PDS libraries are excluded from processing. Only program objects in PDSE libraries are processed.

Note: In a production environment, it is anticipated that only the SCANDIR verb is used, as it provides the necessary data to perform software identification with the minimum resource consumption.

Syntax of Inquisitor requests

This section details the syntax rules and values of request control statements input into the Inquisitor.

Syntax rules are as follows:

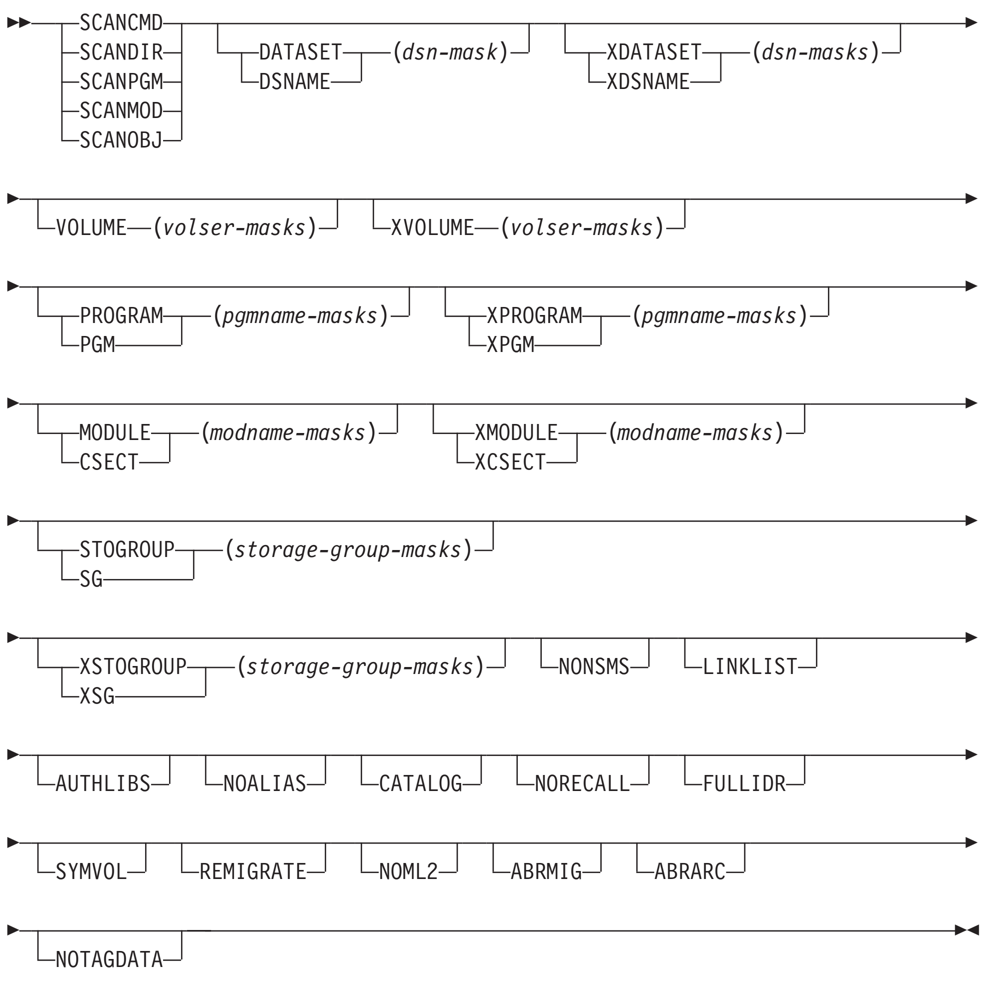
- Only the first 72 bytes of an input record are ever scanned.
- Short records are extended to 72 bytes with blanks, if necessary.
- A continuation on to the next record is requested by a plus or dash.
- A continuation cannot be requested in the middle of a word or value.
- The part of the record following a continuation character is ignored and can be used for comments.
- Blanks, commas, parentheses, and continuation characters, are the only delimiters.

- Blanks and commas are equivalent.
- Records beginning with an asterisk are comment records.
- Records containing only blanks or commas are comment records.
- Comment records are ignored by syntax parsing logic, and do not alter continuation status.
- If the input record contains an ampersand, the system symbol substitution routine ASASYMBM is called to perform symbol substitution processing.
- All input requests are parsed and stored before the first request is processed. If a syntax error is encountered, no requests are processed. This is to reduce the instance of incorrect or unproductive requests triggering lengthy DASD subsystem scans. The error is in the last record echoed in SYSPRINT.
- TSO conventions apply to abbreviations. That is, operands can be abbreviated to the minimum unambiguous length. Verbs cannot be abbreviated.
- Subparameters of value operands are specified in parentheses.
- Value masks are character strings which are compared to data found at run time. Comparison is performed one byte at a time, from left to right. For a match, the characters must compare equal, unless a generic mask character is found.
- System static symbols can be used to construct value masks.
- Valid generic mask characters are a percent (%), to flag a match for any single character, and an asterisk (*), to flag a match for any character string segment of zero or greater length.

Control statement specification

This syntax diagram details Inquisitor request verbs and operands:

Syntax (Inquisitor request verbs and operands)



Defaults are:

- **DSNAME(*) VOLUME(*) PROGRAM(*)**
- No operands are required.
- All operands are optional.

Possible operands:

DATASET Alias: DSNAME

This operand specifies one or more 1 to 44 byte data set name masks. Only data sets with names matching any masks specified here are processed. Data sets with names not matching any masks specified here are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for selection matching. The precise treatment of asterisks in these masks is altered by the presence of the CATALOG keyword in the request. When CATALOG is specified, mask matching becomes qualifier aware and a single asterisk represents one, or part of, one qualifier only. When CATALOG is specified, use a double asterisk to specify any number of qualifiers. The data set name selection mask is the only mask affected by the CATALOG keyword. When the CATALOG keyword is present, exactly one DSNAME mask must be specified.

XDATASET Alias: XDSNAME
This operand specifies one or more 1 to 44 byte data set name masks. Data sets with names matching any mask specified here are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for exclusion matching. If this operand is used, each mask must specify a subset of a DATASET mask.

VOLUME
This operand specifies one or more 1 to 6 byte volume serial number masks. Only volumes with serial numbers matching any mask specified here are processed. Volumes with serial numbers not matching any mask specified here, are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for selection matching. A volume serial number mask of six asterisks specifies the current IPL volume, which is ascertained during execution.

XVOLUME
This operand specifies one or more 1 to 6 byte volume serial number masks. Volumes with serial numbers matching any mask specified here are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for exclusion matching. If this operand is used, each mask must specify a subset of a VOLUME mask. A volume serial number mask of six asterisks specifies the current IPL volume, which is ascertained during execution.

PROGRAM Alias: PGM
This operand specifies one or more 1 to 8 byte program name masks. Only programs with names matching any mask specified here are processed. Programs with names not matching any mask specified here, are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for selection matching.

XPROGRAM Alias: XPGM
This operand specifies one or more 1 to 8 byte program name masks. Programs with names matching any mask specified here are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for exclusion matching. If this operand is used, each mask must specify a subset of a PROGRAM mask.

MODULE Alias: CSECT
This operand specifies one or more 1 to 8 byte module name masks. The presence of this operand means that extra information about program routines is collected. Only CSECTs with names matching at least one mask encountered in processed programs can have this extra information collected. Multiple masks must be separated by one or more delimiters.

This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for selection matching. This operand is ignored for a SCANDIR request.

XMODULE Alias: XCSECT
: This operand specifies one or more 1 to 8 byte module name masks. CSECTs with names matching any mask specified here cannot have extra information collected. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for exclusion matching. If this operand is used, each mask must specify a subset of a MODULE mask. This operand is ignored for a SCANDIR request.

STOGROUP Alias: SG
: This operand specifies one or more 1 to 8 byte storage group name masks. SMS-managed volumes in a storage group with a name matching any mask specified here are processed. SMS-managed volumes in a storage group with a name that does not match any mask specified here, are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for selection matching. Volumes which are not SMS-managed are not processed unless the NONSMS keyword operand is specified.

XSTOGROUP Alias: XSG
: his operand specifies one or more 1 to 8 byte storage group name masks. SMS-managed volumes in a storage group with a name matching any mask specified here are not processed. Multiple masks must be separated by one or more delimiters. This operand can be specified more than once in a request, whereupon all masks specified in all occurrences of this operand are checked for exclusion matching. If both this mask and a STOGROUP mask are used, then each mask must specify a subset of a STOGROUP mask.

NONSMS
: This keyword operand specifies that volumes which are not SMS-managed are eligible for processing. The presence of this operand means that SMS-managed volumes are not processed unless the STOGROUP operand was used to supply a storage group name mask.

LINKLIST
: This keyword operand specifies that all link list data sets are to be unconditionally included for processing.

AUTHLIBS
: This keyword operand specifies that all APF authorized data sets are to be unconditionally included for processing.

NOALIAS
: This keyword operand specifies that any program member marked as an alias is to be excluded from processing.

CATALOG
: This keyword operand specifies that data sets to be processed are located from a catalog search rather than VTOC searches. Data set alias names are not processed. The Inquisitor triggers and waits for a RECALL operation for each migrated data set which passes data set name mask processing, unless NORECALL is also specified.

NORECALL
: This keyword specifies that migrated data sets are not to be recalled and are excluded from processing. This operand only has effect when the CATALOG operand is also specified. Data sets with a catalog entry indicating a volume serial number of MIGRAT, or ARCIVE, are deemed to be migrated.

FULLIDR
: This keyword operand specifies that a full scan of CESD and IDR records is to be performed, even when a module would not have been selected for such processing. Depending upon the exact nature of the request being run, this operand can significantly elongate the elapsed time of Inquisitor runtime.

  This operand is ignored for a SCANDIR request.

SYMVOL
: This keyword operand specifies that when a load library resides on the IPL volume, or on a volume with a serial number which matches the value of a system static symbol, then the output does not contain the actual volume serial number, but six asterisks for the IPL volume or the symbol name for other volumes. Only symbols with names which are six characters long, including the leading ampersand and excluding the trailing period, are considered for this processing.

  If you use this keyword to collect your data, then you must also use the corresponding setting SYM(Y) in the Usage Monitor. You cannot use the SYMVOL parameter in the Inquisitor without using SYM(Y) in the Usage Monitor. By doing this, the Inquisitor data matches the Usage data, otherwise the Usage data is not imported to the correct products.

REMIGRATE
: This keyword operand specifies that when a data set which had to be recalled has been processed, HSM is requested to migrate the data set again asynchronously. Migrated data sets can only be processed when the CATALOG operand is also specified. Only data sets with a catalog entry indicating a volume of MIGRAT are remigrated.

  The presence of this operand requires that the MCDS file is allocated to the HSM MCDS. Access to the MCDS allows the Inquisitor to avoid recalls for data sets which are not partitioned, do not have an undefined record format, and do not have a block size of at least 1024.

NOML2
: This keyword operand specifies that data sets migrated to level two are not to be recalled and are excluded from processing. Migrated data sets can only be processed when the CATALOG operand is also specified. Only data sets with a catalog entry indicating a volume of MIGRAT are checked for level two status.

  The presence of this operand requires that the MCDS file is allocated to the HSM MCDS. Access to the MCDS allows the Inquisitor to avoid recalls for data sets which are not partitioned, do not have an undefined record format, and do not have a block size of at least 1024.

ABRMIG
: This keyword operand indicates that when a catalog entry with a volume of MIGRAT is encountered, the FDRABR product is to be invoked to

determine whether a recallable archived copy of the data sets is available or not. If it is, then the data set is processed. If not, then the data set is not processed.

The NORECALL operand takes precedence over this operand.

The effect of ABRMIG is not affected by the ABRARC operand.

The presence of this operand requires that the ABRIN and ABRPRINT files are allocated.

ABRARC
This keyword indicates that, when a cataloged data set cannot be found on the volume, the FDRABR product is to be invoked in order to determine whether a recallable archived copy of the data set is available. If it is, then the data set is processed. If not, the data set is not processed.

The NORECALL operand takes precedence over this operand.

The effect of ABRARC is not affected by the ABRMIG operand.

The presence of this operand requires that the ABRIN and ABRPRINT files are allocated.

NOTAGDATA
This keyword indicates that data written to program libraries by the Product Tagger is not to be collected and written to the Inquisitor output file. Use this operand only when you do not want to update the Local Knowledge Base during the import process with the latest Tagger data that could be found by the Inquisitor.

Examples

These examples are provided to illustrate some possible scenarios where the scope and type of processing is customized. The first three examples are equivalent, and request data collection for all programs on all online DASD volumes.

Example 1:

```
SCANDIR
SCANDIR DA(*) PGM(*)
SCANDIR VOL(*) DS(*)
```

Example 2:

To scan all SMS-managed volumes except volumes in storage group SGWORK use:

```
SCANDIR STOGROUP(*) XSTOGROUP(SGWORK)
```

Example 3:

To scan all volumes except volumes in storage groups with names beginning with SGW use:

```
SCANDIR XSTOGROUP(SGW*)
```

Example 4:

To scan all volumes with serial numbers beginning with TSO and WRK, these two requests are used in a single program run:

```
SCANDIR VOLUME(TSO*)
SCANDIR VOLUME(WRK*)
```

Example 5:

To scan all volumes except those with serial numbers beginning with TSO and WRK use:

```
SCANDIR XVOLUME(TSO* WRK*)
```

Example 6:

To scan all volumes with serial numbers beginning with USR (which are also in SMS storage groups with names beginning with SG) and programs with names beginning with UTIL, and to collect extra information about CSECTs with names beginning with DATE, except DATENCOD, use: .

```
SCANDIR VOLUME(USR*) STOGROUP(SG*)  +
PROGRAM(UTIL*) MODULE(DATE*) XMODULE(DATENCOD)
```

Example 7:

To scan all data sets with high level qualifiers of SYS1, SYS2, SYS3, except z/OS distribution libraries, use:

```
SCANDIR DSNAME(SYS%.*) XDSNAME(SYS1.A*)
```

Example 8:

To restrict this example's data to cataloged data sets, use:

```
SCANDIR DSNAME(SYS%.**) XDSNAME(SYS1.A*) CATALOG
```

Note: Note the extra asterisk in the data set name selection mask. Without this, only data set names with two qualifiers are selected. Data set name exclusion processing is not changed by the CATALOG operand.

Example 9:

To scan the current IPL volume, and any other link, list, and APF authorized libraries use:

```
SCANDIR VOLUME(******) LINKLIST AUTHLIBS
```

Example 10:

To scan the single cataloged data set SYS1.PPLIB without a lengthy DASD subsystem scan use:

```
SCANDIR DATASET(SYS1.PPLIB) CATALOG
```

Example 11:

To scan all cataloged SYS1 and SYS2 data sets use (a) two requests in a single program run, or (b) a single request. The two approaches exhibit similar resource consumption:

```
SCANDIR DA(SYS1.**) CAT

SCANDIR DA(SYS2.**) CAT
SCANDIRDS(SYS%.**) CAT XDSN(SYS3.*,SYS4.*,SYSA.*)
```

Where the XDSN operand lists all 4-character high-level qualifiers in the system beginning with SYS except for SYS1 and SYS2.

Note: SCANDIR DS(SYS1.**,SYS2.**) CAT is not allowed.

Example 12:

These examples are all equivalent. They scan the entire DASD subsystem for all data sets with a first qualifier of SYS1 or SYS2, excluding those with a second qualifier beginning with A.

(a)

```
SCANDIR DA(SYS1.*,SYS2.*) XDA(SYS1.A*,SYS2.A*)
```

(b)

```
SCANDIR DA(SYS1.*    +
SYS2.*)   +
XDA(SYS1.A*   +
SYS2.A*)
```

(c)

```
SCANDIR DA(SYS1.*)   +
DA(SYS2.*)   +
XDA(SYS1.A*)  +
XDA(SYS2.A*)
```

(d)

```
SCANDIR DA(SYS1.*)  XDA(SYS1.A*)  +
DA(SYS2.*)  XDA(SYS2.A*)
```

(e)

```
SCANDIR DA(SYS1.*)  XDSN(SYS1.A* SYS2.A*)  DS(SYS2.*)
```

Example 13:

Here, the entire DASD subsystem is processed, but the volume serial numbers are replaced in the output data by the name of a corresponding symbol, if one is defined, or by six asterisks for the IPL volume.

```
SCANDIR SYMVOL
```

The SYMVOL operand must be used in conjunction with the SYM command of the Usage Monitor otherwise the Usage data does not match the Inquisitor data.

Designing requests

For an individual Inquisitor request, a given volume is never scanned more than once. When a series of requests is based upon separate groups of volumes, using volume serial number or SMS storage group selection criteria, it also follows that a volume is not scanned more than once. However, when a series of requests for separate groups of data sets based solely upon data set name selection criteria is made, a volume is scanned once for each request.

As a result, it might be preferable to combine all selection and exclusion criteria into a single request to ensure that a volume is scanned only once. This consolidation of numerous criteria into a single request is not always possible if data sets are to be located by catalog search. This is because the use of the

CATALOG keyword prevents the specification of multiple data set name selection masks. However, if catalog search is used, then the issue of scanning any VTOC more than once does not arise.

A volume with no program libraries is processed relatively quickly. Alternatively, for systems with large numbers of volumes known not to have any program libraries of interest, processing duration is noticeably reduced by excluding such volumes from an Inquisitor scan.

For example, volumes exclusively containing databases, or temporary data sets, do not have any files suitable for Inquisitor processing, but the VTOCs of those volumes are still read unless excluded by the appropriate selection criteria.

To illustrate this further, consider a system with these DASD subsystem usage elements:

System platform
Non-SMS and storage group SYSTEM

Work pool
Storage group TEMP containing temporary and short-lived (two days) permanent files

TSO Storage groups TSOONE and TSOTWO

Non-DB application
Non-SMS and storage groups BATCH1 and BATCH2

Databases
Non-SMS volumes DBA001 to DBA099 and SMS storage groups DB01, DB02 and DB03

This list makes these assumptions:

- No need for data from libraries which does not exist for more than two days.
- No program libraries on database volumes.
- Applications combine their program libraries and non-DB files.
- TSO users can have program libraries.
- Management requires information regarding all potentially permanent executable software.

To acquire Inquisitor data from all useful sources without processing volumes more than once, and without processing irrelevant volumes, these requests could be specified in a single Inquisitor run. For example:

```
SCANDIR SG(SYSTEM)
SCANDIR SG(TSO*)
SCANDIR SG(BATCH*)
SCANDIR NONSMS XVOL(DB*)
```

This is consolidated into a single request giving the same result.

```
SCANDIR SG(SYSTEM TSO* BATCH*) NONSMS XVOL (DB*)
```

Processing migrated libraries

For systems using IBM's Hierarchical Storage Manager data management product DFSMShsm, referred to here as HSM, to perform space management, several request keywords are of interest when performing CATALOG requests.

Unless CATALOG is specified, only program libraries on primary volumes are processed by the Inquisitor.

With the CATALOG operand specified, migrated program libraries are eligible for processing, unless the NORECALL operand is also coded. Specifying NORECALL causes libraries cataloged on volumes MIGRAT or ARCHIVE (if no MCDS DD statement is present) to be considered as migrated by the Inquisitor. These are consequently not processed, thereby avoiding their recall to primary volumes.

The presence of a usable HSM MCDS data set allocated to the MCDS file informs the Inquisitor the system has HSM installed. As a consequence, the data sets cataloged on a volume called ARCIVE are not considered to be migrated.

When REMIGRATE or NOML2 is specified, the HSM MCDS is required to be accessible by the MCDS JCL DD statement. No combination of REMIGRATE, NOML2, and NORECALL is considered an error.

When NORECALL is not specified, and REMIGRATE or NOML2 is specified, the MCDS is read to fetch the details of data sets cataloged on volume MIGRAT. ARCIVE is treated as a primary volume. A recall for Inquisitor processing only occurs if the MCDS record indicates that the data set is partitioned with an undefined record format and a block size of at least 1024.

If a specific data set record cannot be read from the MCDS, then it is assumed that the migrated data set does not exist and it is not processed.

The NOML2 keyword forces the exclusion of data sets migrated to level 2.

The REMIGRATE keyword specifies that after a recalled data set has been processed by the Inquisitor, the Inquisitor requests HSM to remigrate the data set. The Inquisitor does not wait for the migration, but begins to process the next data set immediately after making the request to HSM. Migration level two is never specified by the Inquisitor for the migration, even if the data set was recalled from ML2. However, it might be selected by HSM as a result of SMS management class settings.

If none of the NORECALL, REMIGRATE, and NOML2 keywords are present for a catalog request, then the HSM MCDS is not required to be allocated. However, if the MCDS file is allocated, the Inquisitor can use it to avoid recalling data sets which are not program libraries. If the MCDS file is not present, a CATALOG request without the NORECALL operand can trigger recalls for data sets of any data set organization and record format. In all cases, generation data sets are not recalled for Inquisitor processing.

Collection of Inquisitor output

The output from the Inquisitor is listed here as :
- Spooled output including the job log and SYSPRINT file.
- The sequential file containing the extracted information.

The SYSPRINT file contains statistics of processing carried out by the Inquisitor, and the status on any errors encountered.

A program parameter of PGMMSG is specified to request messages reporting member-level anomalies found by the Inquisitor.

All SYSOUT data sets created by the job, including the job log, JCL, and message streams accumulated by the system, form a key resource in problem determination. If there is a problem with the Inquisitor, these files are required in order to diagnose the problem.

The creation of an output file is the main purpose of an Inquisitor execution run. It contains detailed information about programs encountered, as well as audit data which is used in problem diagnosis.

As the size of the HSIPOUT data set might be 200MB or more, it is recommended that the HSIPOUT file is replaced by the HSIPZIP file. HSIPZIP data is HSIPOUT data that has been translated from EBCDIC to ASCII, and compressed by the ZIP process. This reduces the space and time required on z/OS to contain and transfer the file.

Using a compressed form of the data set not only reduces the space used on the DASD, but reduces the transfer time when the file needs to be transmitted to other systems.

The ZIP process used by the Inquisitor requires some update-in-place processing, so the HSIPZIP file must be allocated to a DASD sequential data set which is not compressible. To collect the output in a compressible, extended-format sequential data set, allocate the data set to the HSIPOUT file.

Inquisitor for z/OS UNIX

The Inquisitor for z/OS UNIX is a batch job which collects information about executable software existing in HFS and zFS data sets currently mounted and accessible to z/OS UNIX. The resulting file is then post processed for data import.

The Inquisitor for z/OS UNIX produces a set of record types which is different from that produced by the Inquisitor for z/OS. Nonetheless, both jobs collect the same types of information about executable programs.

The Inquisitor for z/OS UNIX processes the HFS root directory, as well as all subdirectories. For this reason, it needs to run with a UID, which allows access to all directories and programs which need to be examined. If the Inquisitor for z/OS UNIX does not have permission to access a directory, then no information is collected from that directory, or any of its subdirectories.

The HSIXROOT file is used to nominate one or more directories to be considered root directories. When specified, only the nominated directories and their subdirectories are processed. This facility is useful when only a subset of the file hierarchy needs to be scanned.

Apart from the HSIXROOT file, there are no inputs which can restrict, alter, or filter the type of processing to be carried out. Unlike the Inquisitor for partitioned data sets, the Inquisitor for z/OS UNIX does not have any control statements.

Inquisitor JCL for z/OS UNIX

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Inquisitor for z/OS UNIX. Members beginning with UIQ and suffixed with the Inquisitor schema name contain JCL to run the Inquisitor for a given schema.

Run time for this job depends on the size and complexity of the UNIX directory structure to be scanned. It is recommended to run this job during off-peak periods.

Note: In order to collect all relevant z/OS UNIX data, the user running the Inquisitor for z/OS UNIX must have access to all HFS directories, including the root. This ensures that all z/OS UNIX data is collected.

Program parameter details are:

- If a program parameter is specified, it must start with a slash.
- If more than one parameter setting is specified, the settings must be separated by a comma.
- If a message level setting keyword of PTHMSG, PGMMSG, or ALLMSG is specified, it must be at the start of the parameter after the slash.
- PTHMSG requests that a message is written to HSIXMSG each time a directory is opened and closed.
- PGMMSG requests that the processing of every program is logged to the HSIXMSG report.
- ALLMSG requests both PTHMSG and PGMMSG message logging.
- The SID value is up to four characters long and specifies the system identifier to be contained in the Inquisitor's output data.
- If the SID identifier override is omitted, then the system SMF identifier is used. The SID parameter setting is for use when the SMF system identifier of a system is not unique.
- The RUN value is up to eight characters long and specifies an optional run identifier which is contained in the output header record.
- The PLX parameter is used to identify if the Inquisitor data being collected is a SYSPLEX. The value is either Y or N. If the PLX parameter is not used, then the default value of N is created in the Inquisitor header record. For example:
  ```
  Syntax: PLX=Y | N
  ```
- The parameter OUT is used to control the format of the output files:

  OUT=Z
  Zipped output requiring ddname HSIXZIP

  OUT=T
  Text output requiring ddname HSIXOUT

  OUT=B
  Both zipped and text output requiring both ddnames

  The default is OUT=Z

The program parameter is omitted from the JCL.

Modify the STEPLIB DSNAME statement with the correct high-level qualifier so that it points to the correct load library.

Ensure the HSIXZIP output file has an appropriate data set name specified, and that the UNIT and SPACE are correct.

Files used by the Inquisitor for z/OS UNIX

UNIX System Services for z/OS uses the Hierarchical File System (HFS) to store the physical files it processes. Even though HFS files are housed in data sets which are cataloged and have VTOC entries, an individual HFS or zFS data set cannot be accessed by the usual access methods. A single HFS or zFS data set contains an entire structure, consisting of a root directory or a hierarchy of subdirectories, with any one directory containing any number of separate files.

HFS files are designed to be accessed by UNIX applications. A program is usually an individual HFS file. The Inquisitor for z/OS UNIX is the product component used to discover UNIX programs in the z/OS UNIX file system.

The files needed to run the Inquisitor for z/OS UNIX are:

HSIXMSG
Report file used by HSIXINQ.

SYSPRINT
Used by Language Environment (LE), which is required to be in the standard module search path.

SYSOUT
Used by Language Environment (LE), which is required to be in the standard module search path.

HSIXZIP
The output file containing compressed Inquisitor for z/OS UNIX data. It is written using a variable length record format. DCB information must be provided to ensure optimal use of DASD space.

- For a 3390, the use of RECFM=VB,LRECL=27994,BLKSIZE=27998 is advised.
- For a 3380, the use of RECFM=VB,LRECL=23472,BLKSIZE=23476 is advised.
-

The HSIXZIP file must never undergo any translation when being transferred, whatever the architecture of the target system. That is, only BINARY transfers are to be used to transport the file.

HSIXOUT
The output file containing uncompressed Inquisitor for z/OS UNIX data, although the use of HSIXZIP is preferred due to its reduced space requirements. HSIXOUT also contains variable length records. The program supplies the appropriate LRECL. By default, system determined block size is used.

HSIXROOT
HSIXROOT can contain one or more optional 80 byte records, each of which specifies a directory path to be considered, and a root directory to be processed. If HSIXROOT is empty, then a slash (/) is considered to be the only root directory to be processed. Any HSIXROOT record that is present must contain a path name beginning with a slash in column one, and end with one or more blanks or the end of the record. If the specified path does not end in a slash, one is added. HSIXROOT records with a blank in column one are discarded.

The points made in the section discussing the Inquisitor output files and the update-in-place file I/O performed by the ZIP process, and its consequences for compressible files, also apply to the Inquisitor z/OS UNIX output files.

Security considerations when running the Inquisitor against z/OS UNIX files

To allow the Inquisitor Batch Job unrestricted read access to all z/OS UNIX files, consider using the UNIXPRIV RACF Resource Class which alleviates the need for UID(0).

This sample definition could be used by your Security Administrator to Define, Permit, Activate, and RACLIST the RACF UNIXPRIV Class:

```
RDEL UNIXPRIV SUPERUSER.FILESYS.**
RDEF UNIXPRIV SUPERUSER.FILESYS.** UACC(NONE) OWNER(MACNIVE)
PE  SUPERUSER.FILESYS.** CLASS(UNIXPRIV) RESET
PE  SUPERUSER.FILESYS.** CLASS(UNIXPRIV) ID(JKATNIC) ACCESS(READ)
SETR CLASSACT(UNIXPRIV)
SETR RACLIST(UNIXPRIV)
SETR RACLIST(UNIXPRIV) REFR
```

Chapter 9. Usage Monitor

The Usage Monitor is a server address space which runs as a started task. Work is queued to it from all address spaces where programs are used. The Usage Monitor address space needs to be given a reasonably high dispatching priority. While it would be safe to run it at the highest possible dispatching priority, you might choose to give it a priority below those of other high priority workloads, such as z/OS monitors, IRLM, and VTAM®. It is important to note that it needs to be dispatched frequently to process work queue elements in a timely manner.

Two different types of work can run in the server address space:

1. Moving of captured data into the data space repository. This is CPU bound, but of short duration.
2. Writing of the accumulated program Usage data from the data space to a sequential file by the writer task. This is usually I/O bound.

The Usage Monitor runs APF authorized and is nonswappable.

Usage Monitor JCL

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Usage Monitor. Member HSISUMON contains JCL to run the Usage Monitor.

Parameters for the Usage Monitor job are found in PARMLIB member HSISMNPM.

Files used by the Usage Monitor

Apart from files normally associated with running a background process under z/OS, the Usage Monitor has three product specific files which must be allocated in the task JCL. They are:

HSIZIN

A sequential file consisting of fixed length 80 byte records. It contains initial commands which are run before data collection becomes active. This file must contain the data set prefix to be used for dynamically created output files. This prefix can be changed subsequently by an operator MODIFY command.

HSIZIN is opened, read, and closed during initialization processing. Do not specify FREE=CLOSE in the JCL for HSIZIN, or refresh processing is not possible.

Data set UM.HLQIDS, containing high-level qualifier listing, is concatenated to HSIZIN.

HSIZMSG

The report file. The report contains the initial commands issued and indicates their degree of success, as well as regular status reports, refresh reports when appropriate, and a termination report. It consists of fixed length 121 byte records. The ANSI carriage control is used. You should consider keeping the report file with the Usage Monitor MSGCLASS output in order to simplify diagnostic procedures.

SYSOUT
A report file used by the SORT program. Its attributes are determined by the SORT program. It is not used unless sorted output data is requested. The SRT command controls whether or not output is sorted. Sorted data is imported into the appropriate databases more quickly than unsorted data.

Output files containing program Usage data are dynamically allocated by the Usage Monitor. The data set name prefix, the allocation unit, and the primary and secondary space allocation quantities (in tracks) need to be customized for the target system.

Output data sets have names with Dyyyyddd.Thhmmsst appended to the specified prefix. The yyyyddd is the Julian calendar date, and hhmmsst is the time to tenths of a second that the writer task is initiated.

The use of exclusion masks to reduce data

The data from a significant number of program usage events does not contribute meaningfully to the task of managing the software inventory. To reduce the processing of this unnecessary data, two mechanisms which allow some data to be excluded from collection have been provided. They are exclusion masking based on program name, and exclusion masking based on data set name.

Program name exclusion masking

A program name exclusion table exists which contains program name masks. When a program usage event is detected by the Usage Monitor, the program name is checked against entries in the program name exclusion table. When a match is found, the usage event data is discarded. No further processing for that event is performed by Tivoli Asset Discovery for z/OS.

Each table entry contains a program name compare string up to 8 bytes long. The string is either an 8 byte program name, or a shorter program name prefix. When entering these strings with the EXC command, a prefix is denoted by using an asterisk as the last character.

The program name exclusion table resides in key zero common storage, and its size is always a multiple of 4,096 bytes. The minimum table size can house up to 339 entries, and the table size increases dynamically, as required. The default program name exclusion table contains entries to exclude data pertaining to usage of many programs which are part of the operating system.

In order to add, reset, remove, or display the entries to the tables, use these commands:

EXC To add entries to the program name exclusion table, or to reset the table to its default contents.

DEL To remove some, or all, entries from the table.

D-X To display the current contents of the table.

Data set name exclusion masking

Once the Usage Monitor has ascertained the name of the data set from which a used program was fetched, the data set name is used to decide if the Usage data for the event is retained for collection or discarded. To perform this process, two lists of data set name masks are scanned; the first is the data set name inclusion mask list, and the second is the data set name exclusion mask list.

To avoid excessive storage and processor resource consumption, it is preferable to keep the number of elements in each list to a minimum. This is achieved by using generic masks to cover many data set names. The inclusion mask list is provided so that specific exceptions to broad exclusion rules may be specified. Unless data set name exclusion is to be used, the inclusion list cannot provide any useful function.

Elements of both lists reside in key zero common storage. Each element occupies 48 bytes, and contains a data set name mask up to 44 bytes in length. The mask can contain a percent sign in any location, which signifies that any character in that position is considered a match. The mask can end with an asterisk, which signifies that no more characters need be tested in the left-to-right comparison scan of the captured data set name and the data set name mask.
- Use the XDS command to add a data set name mask to the exclusion list.
- Use the IDS command to add a data set name mask to the inclusion list.
- Use the XDD command to deactivate a data set name exclusion mask.
- Use the IDD command to deactivate a data set name inclusion mask.
- Use the D-D command to display the currently active masks in both lists.

Both lists have no elements until an XDS or IDS command is processed. Storage is dynamically acquired for each element as required. To ensure system integrity, XDD and IDD commands do not cause the storage of a deactivated element to be freed, but simply mark the element as inactive. When a deactivated mask is reactivated, the existing entry is marked as active without the further acquisition of storage.

When the Usage Monitor address space first initializes, all elements of both lists left in storage from a previous run of the Usage Monitor are freed before the processing of initial commands and the commencement of data collection. There is no requirement to use either data set name mask list at any stage.

Starting the Usage Monitor

A Usage Monitor member named HSIJMON is provided in proclib SHSIPROC. If you want to start HSIJMON as a started task, you need to customize this member according to your site’s requirements, and copy it to an authorized PROCLIB.

To start the Usage Monitor in normal mode issue:
```
S HSIJMON
```

To start it in passive mode issue:
```
S HSIJMON,PARM=PASSIVE
```

Note: PASSIVE is the only valid value of the optional program parameter. PASSIVE mode means that the Usage Monitor is running, but not collecting any Usage data.

Stopping the Usage Monitor

To stop the Usage Monitor, issue any of these commands:

```
P HSIJMON
F HSIJMON,STOP
F HSIJMON,END
```

These commands cause the address space to stop data collection, attach a writer task to process the existing data in the repository, wait for the writer task to complete, sort the data, and then terminate.

```
F HSIJMON,QUICK
```

This command causes the server address space to stop collecting data, attach a writer task to process the existing data in the data space, wait for the writer task to complete, and then terminate without sorting the data. For immediate termination, issue:

```
F HSIJMON,CAN
```

This command causes the server address space to stop data collection, detaches any running writer task which renders the output data set unusable, deletes the current data space without writing out its contents, and terminates.

Refreshing Usage Monitor settings

The Usage Monitor has a range of commands to alter processing, any of which may be issued dynamically. However, these commands only have an effect for the duration of the current Usage Monitor session.

It is often desirable to implement a change, both to the running Usage Monitor and to the initialization commands to be used by subsequent Usage Monitor sessions on startup. The refresh facility is available to assist with the implementation of these permanent changes.

Refresh processing involves the execution of the command stream placed in the HSIZIN file without the requirement of stopping and restarting the Usage Monitor. As a result, refresh processing can verify the validity of the initialization command stream so that changes are made and tested dynamically, ensuring that future Usage Monitor sessions do not encounter initialization command stream errors.

The response to each command in the HSIZIN file is written to the HSIZMSG file. A summary WTO message, indicating whether any errors are found or not, is issued after refresh processing has finished.

Some commands set a switch for logic control, or set a numerical value to be used during some processing. These commands specify the values to be used in the future. Other commands pertaining to inclusion and exclusion masking add a mask to, or remove a mask from, the active mask list, and so are part of an accumulation of commands which specify future processing.

Consider the case where several exclusion masks are active, and a change to deactivate one of the masks is required. A command to deactivate the mask is issued dynamically, but if this change is to be made permanent, then the HSIZIN file needs to be updated. This update would usually consist of deleting the command which specifies the exclusion mask in question.

In this scenario, simply issuing the commands remaining in HSIZIN would not deactivate the mask now omitted from the command stream in the general case. So, to allow such a change to be implemented by deleting a masking command from HSIZIN and performing a refresh, the Usage Monitor performs extra actions before executing the commands in HSIZIN.

Before the first HSIZIN command is run during refresh processing, the program mask exclusion list is set to the default list. Further, all data set name exclusion masks are deactivated, and all data set name inclusion masks are deactivated. This order of deactivation ensures that there is no loss of data that would otherwise be collected. However, there is the possibility that data which would have been excluded is collected during the short window between the reset of the mask lists and the processing of the HSIZIN commands.

Stopping the Usage Monitor and restarting it, produces the same active exclusion masks as a refresh. It also produces a data collection outage. See the section detailing the REF command for a list of the processes performed during a refresh operation.

Usage Monitor commands

The Usage Monitor commands are passed to the Usage Monitor from the HSIZIN input file, or by an operator MODIFY command.

Syntax rules are as follows:
- All commands are three characters long.
- Operands or sub parameters are specified in parentheses.
- Multiple sub parameters are separated by commas.
- The command must not contain any embedded blanks.
- Commands must start in column one.

Details of each command are:

CAP - Set hardware capacity collection status

CAP is used to specify if the Usage Monitor is to output records containing information about the hardware capacity of the system. Collecting this information is important when hardware capacity changes dynamically.

A change to this setting does not take effect until the next data space repository switch.

Syntax

```
►►──CAP(──┬─Y─┬──)──────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that hardware capacity data is collected and output.

N specifies that hardware capacity is not collected or output.

If no CAP command is issued, then hardware capacity data is collected. CAP(Y) is the default.

Example 1

Collect hardware capacity data.

```
F HSIJMON,CAP(Y)
```

Example 2

Do not collect hardware capacity data.

```
F HSIJMON,CAP(N)
```

CSA - Set the (E)CSA queuing storage limit

CSA is used to specify a limit to the quantity of (E)CSA storage used to queue work. If the Usage Monitor address space is not dispatched in a timely fashion, then a large number of work elements can exist concurrently before being processed. Such work is queued in ECSA until it is transferred to the Usage Monitor repository.

If ECSA is exhausted, then CSA is used.

Data from program usage events occurring while this limit has been reached might not be collected.

An active CSA limit setting stays in force unless overridden, even if the Usage Monitor is stopped and restarted.

Syntax

```
►►──CSA(limit)──────────────────────────────────────────────────►◄
```

limit specifies a number of kilobytes from 0 to 200,000.

If no CSA command is issued then CSA(0) is in force. CSA(0) specifies that the Usage Monitor does not attempt to limit the (E)CSA storage used by work elements awaiting processing.

Example 1

Limit queuing in (E)CSA to 50,000KB (almost 50MB).

```
F HSIJMON,CSA(50000)
```

Example 2

Allow no explicit (E)CSA limit for storing queued data.

```
F HSIJMON,CSA(0)
```

D-A - Display output allocation parameters

D-A is used to verify dynamic allocation details to be used in the creation of output data files. The data set name, primary and secondary space quantities, and unit and optional volume serial number are shown.

Syntax

```
►►──D-A─────────────────────────────────────────────────────────►◄
```

It is advantageous to have this command in the HSIZIN file, after the DSN command, to confirm initial allocation values.

Example 1

Display the current dynamic allocation values.

```
F HSIJMON,D-A
```

D-C - Display the counters and statistics

D-C is used to show the Usage Monitor activity and status indicators. The purpose of this command is to assist IBM technical support in problem diagnosis. The meaning of the output generated by this command is not published.

Syntax

```
►►──D-C──────────────────────────────────────────────────────────────►◄
```

Example 1

Display the current value of internal Usage Monitor counters.

```
F HSIJMON,D-C
```

D-D - Display the data set name inclusion/exclusion lists

D-D is used to show the data set name masks in the inclusion list, followed by the data set name masks in the exclusion list.

If the program library of a program is ascertained when the program usage is detected, the inclusion list is scanned for a match. If no match is found, then the exclusion list is scanned for a match. If a match is found then the program usage data is discarded, otherwise it is collected.

Neither list need be populated in order to collect data. The absence of any entries in the exclusion list means that data collection is not affected by program library data set names.

Syntax

```
►►──D-D──────────────────────────────────────────────────────────────►◄
```

It is advantageous to have this command in the HSIZIN file to confirm system settings.

Example 1

Display the current data set name inclusion and exclusion lists.

```
F HSIJMON,D-D
```

D-I - Display the system Identifier

D-I is used to verify the system identifier, which is written in the output header record. This can be altered by the SID command.

Syntax

```
►►──D-I──────────────────────────────────────────────────────────────►◄
```

It is advantageous to have this command in the HSIZIN file to confirm system settings.

Example 1

Display the current the current system identifier used by the Usage Monitor.

```
F HSIJMON,D-I
```

D-S - Display the Status settings

D-S is used to verify several miscellaneous settings. Other commands are used to alter the individual settings, but this command provides a convenient way to list the current values.

Syntax

```
►►──D-S──────────────────────────────────────────────────────────►◄
```

It is advantageous to have this command in the HSIZIN file to confirm system settings.

Example 1

Display the current values of settings which are altered by the CSA, TRG, z/OS UNIX, LPA, PLX, SYM, SRT, UNK, and ZIP commands.

```
F HSIJMON,D-S
```

D-T - Display the automatic switch-and-write Time setting

D-T is used to verify the time-of-day specified for automatic data space switching and consequent writer task creation. When data from after this time-of-day is detected, data collection is automatically switched to a new data space, and write-out of data in the old data space is initiated.

The UTC or GMT switch time is calculated using local time current at data space creation time. That is, the time when a data space will be terminated is set when it is created. Changes to the system local time offset, such as those caused by a change to Daylight Saving Time, do not alter the UTC or GMT time that the current data space will be closed. The time of the switch after the next switch is calculated using the new local time.

Syntax

```
►►──D-T──────────────────────────────────────────────────────────►◄
```

It is advantageous to have this command in the HSIZIN file to confirm system settings.

Example 1

Display the current automatic switch-and-write time setting.

```
F HSIJMON,D-T
```

D-X - Display the active exclude list

D-X is used to display the active program name mask exclude list. Data is not collected for programs with names that match the mask in any active entry in the exclude list.

Syntax

```
►►──D-X──────────────────────────────────────────────────────────►◄
```

It is advantageous to have this command as the last command in the HSIZIN file, so that the initial exclude list is verified.

Example 1

Display the current exclude list entries.

```
F HSIJMON,D-X
```

DCB - Set output DCB attributes

DCB is used to set DCB attributes, which are optimal for a specific device type.

Syntax

```
►►──DCB(──┬─3390─┬──)────────────────────────────────────────────►◄
          ├─3380─┤
          └─UNKN─┘
```

Note: If no DCB command is issued, DCB(3390)is used.

DCB(3390) sets the output DCB to

```
RECFM=VB,LRECL=27994,BLKSIZE=27998
```

Use this when the output device has 3390 compatible geometry.

DCB(3380) sets the output DCB to

```
RECFM=VB,LRECL=23472,BLKSIZE=23476
```

Use this when the output device has 3380 compatible geometry.

DCB(UNKN) sets the output DCB to

```
RECFM=VBS,LRECL=32756,BLKSIZE=0
```

so that the system determines the optimal block size for the device chosen by dynamic allocation. Use this when the output device type is not known until allocation time.

Some third party FTP products do not process a file with RECFM=VBS correctly, even when no records are actually spanned.

DEL - Deleting Program Mask Entries

DEL can be used to remove program name masks from filter tables. Both default and user-added entries can be removed. The required operand specifies one or more program name masks.

Syntax

```
►►──DEL(──┬─mask─────────────────┬─)──────────────────────────────────►◄
          ├─,mask─┬────────────┬─┤
          │       └─,mask...───┘ │
          └─*ALL*────────────────┘
```

mask specifies a one to eight character program name mask. If the mask ends in an asterisk, only characters before the asterisk are compared, otherwise an exact program name is deemed to have been specified.

ALL specifies every currently active mask. This mask cannot be specified with any other mask.

Except for short test periods, it is expected that default exclusion masks such as IGG* remain active.

Example 1

Remove all entries, so that all possible programs are monitored.

```
F HSIJMON,DEL(*ALL*)
```

Example 2

Remove an exclusion mask to allow the monitoring of LE and REXX modules.

```
F HSIJMON,DEL(CEE*,IRX*)
```

Example 3

Remove an exclusion mask to allow the monitoring of the program called CEE.

```
F HSIJMON,DEL(CEE)
```

DSN - Setting the Data Set Name prefix

DSN is used to specify the first part of the data set names used for the output files. The prefix is specified in the required operand.

Symbols are employed in the construction of the data set name prefix. Available symbols include all z/OS static symbols, as well as &SMF, the system's SMF identifier, and &SYSLPAR, the system's logical partition name.

Syntax

```
►►──DSN(dsnpref)───────────────────────────────────────────────────►◄
```

Dsnpref

specifies a 1 to 26 character data set name prefix. It can contain one or more data set qualifiers, and must not end in a period after any symbol substitution.

Example 1

To get output files with names of the form

```
SYS3.HSIz.HSIJMON.Dyyyyddd.Thhmmsst
```

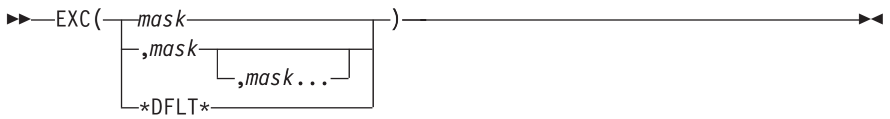
use

```
F HSIJMON,DSN(SYS3.HSIz.HSIJMON)
```

EXC - Adding program mask exclusion entries

EXC is used to add program name masks to the exclusion table. The required operand specifies one or more program name masks.

Syntax



mask specifies a one to eight character program name mask. If the mask ends in an asterisk only, characters before the asterisk are compared. Otherwise, an exact program name is deemed to have been specified.

DFLT
specifies every supplied default entry in the exclusion table is to be made active, and that all user-added entries are to be removed from the exclusion table. This mask cannot be specified with any other mask.

Except for short test periods, it is expected that default exclusion masks such as IGG* would remain active

Example 1

Reset the exclusion table to its default status.

```
F HSIJMON,EXC(*DFLT*)
```

Example 2

Exclude the collection of data for Language Environment modules and REXX modules.

```
F HSIJMON,EXC(CEE*,IRX*)
```

Example 3

Exclude the collection of data for the program CEE.

```
F HSIJMON,EXC(CEE)
```

IDD - Deleting data set name inclusion entries

IDD is used to remove data set name masks previously added by the IDS command.

Syntax

```
►►──IDD(mask)──────────────────────────────────────────────────►◄
```

mask specifies a 1 to 44 character data set name mask. If the mask ends in an asterisk only, characters before the asterisk are compared. Percent signs in the mask indicate that any character in that location is considered a match.

Example 1

Deactivate the SYS3.LINKLIB inclusion mask.

```
F HSIJMON,IDD(SYS3.LINKLIB)
```

IDL - Control Idle Work Element usage

IDL is used to control whether the Usage Monitor allows idle work elements. When the data in a work element has been processed, the element is normally freed in order to return the storage to the system. Allowing idle elements means that processed elements are retained on the idle chain, which is used before acquiring more storage when a new work element is needed.

Allowing idle elements is beneficial as it reduces system storage management overhead. The storage used by idle elements is included in the storage limit set by the CSA command.

Syntax

```
►►──IDL(──┬─Y─┬──)─────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies the Usage Monitor will retain processed elements for reuse subject to the CSA limit setting.

N specifies that all processed work elements are to be freed.

If no IDL command is issued, then the idle chain will be used. That is, IDL(Y) is the default setting.

Example 1

Enable use of the idle element chain.

```
F HSIJMON,IDL(Y)
```

Example 2

Prevent further additions to the idle element chain.

```
F HSIJMON,IDL(N)
```

IDS - Adding Data Set Name inclusion entries

IDS is used to supply data set name masks, which specify data set names to be excluded from exclusion processing. Program Usage data fetched from data sets with names matching inclusion masks, is collected without reference to the data set name mask exclusion list.

Inclusion masks are only useful if there are active exclusion masks. An inclusion mask is normally expected to match a subset of data set names, which would match an exclusion mask.

Syntax

```
►►──IDS(mask)──────────────────────────────────────────────────►◄
```

mask specifies a 1 to 44 character data set name mask. If the mask ends in an asterisk only characters before the asterisk are compared. Percent signs in the mask indicate that any character in that location is considered a match.

Example 1

Consider the case where the intention is to not collect program Usage data for data sets with a high level qualifier of SYS3, except for SYS3.LINKLIB. SYS3.LINKLIB is the only data set with a high level qualifier of SYS3 for which program Usage data is to be collected. Given an active exclusion mask of SYS3.*, this command would satisfy the requirement.

```
F HSIJMON,IDS(SYS3.LINKLIB)
```

JAC - Set job account collection status

JAC can be used to specify if the Usage Monitor is to consider the account code of jobs significant when aggregating data. The Usage Monitor normally aggregates data based on the program name, the job name, and the user ID. The JAC setting is used to add the job account, truncated after 20 characters, to the aggregation key.

Do not instruct the Usage Monitor to collect and preserve all job account codes if they are not important to the administration of your system. Collecting and preserving job accounts increases data volumes.

A change to this setting will not take effect until the next data space repository switch.

Syntax

```
►►──JAC(─┬─Y─┬─)───────────────────────────────────────────────►◄
         └─N─┘
```

Y specifies that job account codes will be collected and output.

N specifies that job account codes will be ignored.

If no JAC command is issued then job accounts will not be collected. JAC(N) is the default.

Example 1

Collect and preserve job account codes.

```
F HSIJMON,JAC(Y)
```

Example 2

Ignore job account codes.

```
F HSIJMON,JAC(N)
```

LLC - Link List Correction

LLC can be used where sites make a number of dynamic link list changes. This command updates the HSIJMON data to point to the correct load library. Only use this command if you allow dynamic link list updates, which alter the relative concatenation numbers of persisting libraries.

Syntax

```
►►──LLC(──┬─Y─┬──)──────────────────────────────────────────────────►◄
          └─N─┘
```

Y a BLDL is performed at write time by the writer task and, if found, the data set name is overlaid.

N do not check for dynamic list updates.

LPA - Set Link Pack Area program monitoring status

LPA is used to specify whether the monitoring of programs residing in the Link Pack Area (LPA) is to occur or not. All types of LPA are included in this category.

Syntax

```
►►──LPA(──┬─Y─┬──)──────────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that LPA program usage is to be monitored.

N specifies that LPA program usage is not to be monitored.

If no LPA command is issued then LPA program usage data is collected. That is, LPA(Y) is the default setting.

Example 1

Collect usage data about LPA programs.

```
F HSIJMON,LPA(Y)
```

Example 2

Stop the collecting LPA program usage information.

```
F HSIJMON,LPA(N)
```

MOD - Alter the operational Mode

MOD is used to set the Usage Monitor into passive mode where no data is collected, or collect mode where data collection is active.

Syntax

```
►►──MOD(──┬─COL─┬──)──────────────────────────────────────────────►◄
          └─PAS─┘
```

COL specifies collect mode.

PAS specifies passive mode.

If no MOD command is issued the Usage Monitor operates in collect mode.

Example 1

Stop the collection of program usage data.

```
F HSIJMON,MOD(PAS)
```

Example 2

Allow the collection of program usage data.

```
F HSIJMON,MOD(COL)
```

PLX - Alter the Sysplex-Wide software flag

PLX is used to set the sysplex-wide software flag. This flag is copied to the header record of each program usage data output data set. It is used by later database processing to indicate that Usage data is matched against a single software inventory, as all sysplex members share DASD volumes.

Syntax

```
►►──PLX(──┬─Y─┬──)────────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that all software is accessible sysplex-wide.

N specifies that each OS image has its own software inventory.

If no PLX command is issued, then a blank is contained in the header record, which means that the sysplex-wide flag has to be set at data import time.

Example 1

Set the sysplex-wide common software inventory flag.

```
F HSIJMON,PLX(Y)
```

Example 2

Reset the sysplex-wide common software inventory flag.

```
F HSIJMON,PLX(N)
```

PRI - Set the data set space Primary Allocation

PRI is used to specify the primary space allocation quantity in tracks. This is used for output data set allocations.

Syntax

```
►►──PRI(trks)──────────────────────────────────────────────────────►◄
```

trks specifies a number of tracks from 0 to 150,000.

If no PRI command is issued, the primary space allocation is 750 tracks. Tivoli Asset Discovery for z/OS uses the RLSE space allocation attribute.

Example 1

Set the primary space allocation to 900 tracks.

```
F HSIJMON,PRI(900)
```

PRS - Set registered software activity data collection status

PRS is used to specify if the Usage Monitor is to output records containing information about registered software activity. This data contains information about the usage of software which uses the system's Register service, and information about registered product PARMLIB settings.

A change to this setting will not take effect until the next data space repository switch.

Syntax

```
►►──PRS(──┬─Y─┬──)─────────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that registered software information will be collected and output.

N specifies that registered software information will not be collected or output.

If no PRS command is issued, then registered software data will be collected. PRS(Y) is the default.

Example 1

Collect registered software data.

```
F HSIJMON,PRS(Y)
```

Example 2

Do not collect registered software data.

```
F HSIJMON,PRS(N)
```

REF - Refresh Usage Monitor settings

REF is used at any time to reset Usage Monitor settings according to commands in the HSIZIN file, without having to stop and start the Usage Monitor. The detailed results of the refresh operation are written to the HSIZMSG file.

The processes of a refresh operation include:

- Verify that HSIZIN is still allocated.
- Open HSIZIN.

- Set the program exclusion list to the default list.
- Deactivate all data set exclusion list elements.
- Deactivate all data set inclusion list elements.
- Process the commands in HSIZIN.
- Close HSIZIN.
- Issue either HSIZ059I or HSIZ060I, as appropriate.

Syntax

```
►►──REF─────────────────────────────────────────────────────────────►◄
```

Example 1

Change Usage Monitor settings to updated values from HSIZIN.

```
F HSIJMON,REF
```

SEC - Set the data set space secondary allocation

SEC is used to specify the secondary space allocation quantity in tracks. This is used for output data set allocations.

Syntax

```
►►──SEC(trks)───────────────────────────────────────────────────────►◄
```

trks specifies a number of tracks from 0 to 150,000.

If no SEC command is issued, the secondary space allocation is 300 tracks. Tivoli Asset Discovery for z/OS uses the RLSE space allocation attribute.

Example 1

Set the secondary space allocation to 600 tracks.

```
F HSIJMON,SEC(600)
```

SID - Set the Usage Monitor System Identifier

SID is used to override the system identifier contained in the output header record. The SMF system identifier is used as a norm, but an override allows the data from separate systems to be differentiated in all instances where duplicate SMF identifiers are in use. Symbols can be employed in the construction of the system identifier. Available symbols include all z/OS static symbols, as well as &SMF, the system's SMF identifier, and &SYSLPAR, the system's logical partition name.

Syntax

```
►►──SID(sid)────────────────────────────────────────────────────────►◄
```

sid specifies an identifier from 1 to 4 bytes in length.

Example 1

Set the output system identifier to PROD.

```
F HSIJMON,SID(PROD).
```

SIZ - Set the data space repository size

SIZ is used to specify the maximum number of entries that the data space repository can hold.

Syntax

```
►►──SIZ(entries)─────────────────────────────────────────────►◄
```

entries specifies a number of entries from 100 to 6,000,000.

If no SIZ command is issued, a data space capacity of 200,000 entries is used. Each entry occupies 272 bytes. As each data space page has data placed in it for the first time, that page has to be backed physically by the system. When a data space is full, a repository switch is triggered automatically. A repository switch also occurs when data stamped after the switch time is detected. Keep these points in mind when customizing the data space capacity..

Example 1

Set the size of future data spaces to one million entries.

```
F HSIJMON,SIZ(1000000)
```

SRT - Set output data Sort status

SRT is used to specify whether or not the output data is to be sorted to optimize database import . While sorted data can be processed more quickly, the Usage Monitor address space uses more resources if sorting is requested. You need to check the customization of the SORT product for suitability, especially in relation to the dynamic allocation of SORTWORK disk space. If sorting is to be performed, ensure that the SYSOUT DD statement is in the Usage Monitor JCL.

Syntax

```
►►──SRT(──┬─Y─┬─)──────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that the output data is to be sorted.

N specifies that the SORT program is not to be invoked.

If no SRT command is issued, then sorted data is written. That is, SRT(Y) is the default setting.

Example 1

Enable the sorting of the output data.

```
F HSIJMON,SRT(Y)
```

Example 2

Disable the sorting of the output data.

```
F HSIJMON,SRT(N)
```

SWI - Switch to a new data space repository

SWI causes a new data space repository to be created and used for subsequent data collection. The data space current at the time of SWI command has its data contents processed by a writer task.

The SWI command has no operands. The SWI command is invalid in the HSIZIN initial command file. As well as the switch caused by an explicit SWI command, automatic switches occur when a repository becomes full, and when data stamped after the switch time is detected. A SWI command might be rejected if the writer task is busy.

Syntax

```
►►──SWI──────────────────────────────────────────────────────────────►◄
```

Example 1

Manually switch to a new repository.

```
F HSIJMON,SWI
```

SYM - Set the Symbolic volume serial on output switch

SYM allows the logged volume serial number of a load library to be either the actual volume serial number or a symbolic value. When symbolic volume serial numbers are allowed, the IPL volume is always reported as six asterisks. For other volumes, if the serial number is found to match the value of a static system symbol, then the name of the symbol is reported instead of the actual volume serial number. Only symbols with names exactly six characters long are considered for this processing. For the purposes of this discussion, the symbol name includes the leading ampersand (&), but excludes the trailing period (.).

Syntax

```
►►──SYM(──┬─Y─┬──)───────────────────────────────────────────────────►◄
          └─N─┘
```

Y	the actual volume serial number is replaced by a symbolic value.
N	the actual volume serial number is always output.

If no SYM command is issued, then the actual volume serial number collected is always output; SYM(N) is the default setting.

SYM(Y) is usually used in conjunction with the SYMVOL operand of the Inquisitor.

Example 1

Allow the output of symbolic volume serial numbers.

```
F HSIJMON,SYM(Y)
```

Example 2

Disallow the output of symbolic volume serial numbers.

```
F HSIJMON,SYM(N)
```

TRG - Set the cache trigger event count

TRG allows the setting of the program usage event cache trigger. Repository entries with usage counts less than the trigger value which are not cached. When usage events are captured for cached entries, usage of common storage and cross-memory POST processing is avoided.

When the cache is full, no additional entries can be cached. About every two hours, a status report, indicating cache usage, is written to the HSIZMSG file. After this the cache is emptied, if at least half full. The cache is also emptied when the collection for a repository is terminated. That is, when the repository is switched or when the Usage Monitor is shutdown.

The cache trigger value is not to be set too low, or little benefit results. The maximum benefit of the cache occurs when the cache contains the entries which are collecting the most frequent program usage events.

The regular status reports in the HSIZMSG file should be examined to help determine the optimal cache trigger count which is often orders of magnitude larger than the default value.

Syntax

```
►►──TRG(count)──────────────────────────────────────────────────►◄
```

count a number in the range from 0 to 999,999,999.

If no TRG command is issued, then the trigger count is set at 1000. TRG(0) specifies that the cache is to be filled as quickly as possible from the next captured program usage event data.

Example 1

Set the cache trigger event count to twelve hundred.

```
F HSIJMON,TRG(1200)
```

UNK - Set the Unknown event collection switch

UNK is used to specify whether events with incomplete data are to be collected or not. The database content is not affected. Collecting extra data is useful in determining why some usage events are not captured. This needs to be set only when requested by IBM support.

Syntax

```
►►──UNK(──┬─Y─┬──)──────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that the ″unknown″ events are to be collected.

N specifies that the ″unknown″ events are not to be collected.

UNK(N) is the default setting.

Example 1

Start the collection of unknown events.

```
F HSIJMON,UNK(Y)
```

UNM - Set user name collection status

Software security packages, such as RACF, have a name field for each user ID defined to the system. The Usage Monitor collects the user ID (up to 8 characters long), and the contents of the name field (up to 20 characters long),as part of the data collection performed when programs are used. UNM is used to specify whether the names of users collected from the security package will be output. The user ID is always output. This setting is checked by the writer task when the data in a data space repository is being processed for output.

Syntax

```
►►──UNM(──┬─Y─┬──)──────────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that collected user names will be written to the output file.

N specifies that collected user names will be discarded.

If no UNM command is issued, then user names will be collected. That is, UNM(Y) is the default.

Example 1

Transmit user names to Usage Import.

```
F HSIJMON,UNM(Y)
```

Example 2

Prevent user names from appearing in any program usage reports.

```
F HSIJMON,UNM(N)
```

UNT - Set the data set allocation Unit

UNT is used to specify the allocation unit to be used for output data set allocations.

Syntax

```
►►──UNT(unitname)───────────────────────────────────────────────────►◄
```

Unitname
specifies a 1 to 8 character long unit name.

If no UNT command is issued, SYSALLDA is used.

Example 1

Set the allocation unit to WORKDA.

```
F HSIJMON,UNT(WORKDA)
```

USS - Set UNIX Program monitoring status

USS is used to determine whether or not the monitoring of programs retrieved from Hierarchical File System (HFS) files is to occur.

Syntax

```
►►──USS(──┬─Y─┬──)──────────────────────────────────────────────────►◄
          └─N─┘
```

Y programs fetched from HFS files are to be monitored.

N programs fetched from HFS files are not to be monitored.

Note: If no USS command is issued, then programs retrieved from HFS files are not monitored. USS(N) is the default setting.

Example 1

Collect maximal information about z/OS UNIX programs.

```
F HSIJMON,USS(Y)
```

Example 2

Stop collecting HFS program usage information.

```
F HSIJMON,USS(N)
```

VOL - Set the data set allocation Volume

VOL is used to specify the allocation volume to be used for output data set allocations. The explicit nomination of a specific volume is necessary when there are no PUBLIC or STORAGE volumes in the allocation unit pool.

Syntax

```
►►──VOL(volume)─────────────────────────────────────────────────────►◄
```

volume specifies a 1 to 6 character long volume serial number.

If no VOL command is issued, a specific volume is not explicitly requested.

Example 1

Set the allocation volume to SCR001.

```
F HSIJMON,VOL(SCR001)
```

WRT- Set the automatic switch-and-Write time of day

WRT is used to specify a time-of-day to end data collection for the current data space, and automatically switch to a new data space. The data write-out for the closed data space is also initiated. These events are triggered when data from after the specified time is detected.

The UTC or GMT switch time is calculated using local time current at data space creation time. That is, the time that a data space will be terminated is set when it is created. Changes to the system local time offset, such as those caused by a change to Daylight Saving Time status, do not alter the UTC or GMT time that the current data space will be closed. The time of the switch after the next switch will be calculated using the new local time.

Syntax

```
►►──WRT(hhmm)──────────────────────────────────────────────────►◄
```

hhmm specifies a 24-hour time-of-day in hour and minute notation. The value must be four (4) decimal digits. The first two digits (hh) must be in the 00 to 23 range. The last two digits (mm) must be in the 00 to 59 range.

If no WRT command is issued, the automatic switch time of midnight is used. That is, WRT(0000) is the default.

Example 1

Set the automatic switch-and-write time to noon.

```
F HSIJMON,WRT(1200)
```

Example 2

Set the automatic switch-and-write time to ten minutes before midnight.

```
F HSIJMON,WRT(2350)
```

XDD - Deleting data set name Exclusion entries

XDD is used to remove data set name masks which were added by the XDS command.

Syntax

```
►►──XDD(mask)──────────────────────────────────────────────────►◄
```

mask specifies a 1 to 44 character data set name mask. If the mask ends in an asterisk, then only characters before the asterisk are compared. Percent signs in the mask indicate that any character in that location is considered a match.

Example 1

Deactivate the SYS3.* exclusion mask.

```
F HSIJMON,IDD(SYS3.*)
```

XDS - Adding data set name Exclusion entries

XDS is used to supply data set name masks which specify data set names to be excluded from data collection. Program Usage data for programs fetched from data sets with names matching exclusion masks is discarded.

When the captured data set name has been matched to an inclusion mask, set by the IDS command, the data is collected without reference to the exclusion mask list.

Syntax

```
►►──XDS(mask)──────────────────────────────────────────────────►◄
```

mask specifies a 1 to 44 character data set name mask. If the mask ends in an asterisk, only characters before the asterisk are compared. Percent signs in the mask indicate that any character in that location is considered a match.

Example 1

Exclude program Usage data from collection for programs fetched from data sets with a high level qualifier.

```
F HSIJMON,XDS(SYS3.*)
```

ZIP - Set the compressed output data switch

ZIP is used to control whether the writer task is to compress output data or not. Compressing the output data reduces data volumes which subsequently reduces data transfer time and storage space requirements.

Syntax

```
►►──ZIP(──┬─Y─┬──)─────────────────────────────────────────────►◄
          └─N─┘
```

Y specifies that output data is to be compressed.

N specifies that output data is not to be compressed.

If no ZIP command is issued, then compressed data is output. That is, ZIP(Y) is the default setting.

Example 1

Ensure that compressed data is being output.

```
F HSIJMON,ZIP(Y)
```

Chapter 10. Inquisitor Import

Inquisitor Import is the process where the Inquisitor data collected previously in jobs beginning with ZIQ (UIQ for z/OS UNIX), and suffixed with the schema name, is imported into DB2 Inquisitor tables.

The three tables TSYSTEM, TLIBRARY, and TMODULE, are the core Inquisitor tables created by DI*iqschemas*

IQ*schema.***TSYSTEM**
: A one-row table with system details.

IQ*schema.***TLIBRARY**
: A small to medium-sized table with one row for each load library, and a row length of 185 bytes.

IQ*schema.***TMODULE**
: A larger table with one row for each load module, and a row length of between 398 and 652 bytes.

Choosing a good naming convention for IQSCHEMAS is very important. As the IQSCHEMAS values are used in creating schema names and table spaces in DB2, there are certain restrictions. The value must have no special characters, and be less than, or equal to, five alphanumeric characters. The first character must be an alphabetic character.

For example, IQSCHEMAS=FF888 would result in creating these DB2 objects:

```
tablespace WSFF888
tablespace WSUF888 for z/OS UNIX
table FF888.TSYSTEM
table UFF888.TSYSTEM for z/OS UNIX
```

Running Inquisitor Import

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to import the Inquisitor data into DB2. Members beginning with ZIM (UIM for z/OS UNIX), and suffixed with the schema name contain JCL to run the Inquisitor Import function for a given schema.

Run times for these jobs depend on the number of modules to be imported into the DB2 Inquisitor tables. It is recommended to run these jobs during off-peak periods.

Note: Data created from the previous Inquisitor import has to be deleted. For performance reasons, Inquisitor tables are dropped and then recreated when this job is run.

TPARAM parameters

SSID= DB2 subsystem name. Value assigned, as defined in job HSISCUST.

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

DATABASE=
: Name of database. Value assigned, as defined in job HSISCUST.

IQSCHEMA=
: Inquisitor schemas. Value assigned, as defined in job HSISCUST.

GKBSCHEMA=
: Default is GKB7. Global Knowledge Base schema. No value specified for Inquisitor Import for z/OS UNIX.

LKBSCHEMA=
: Default is LKB7. Local Knowledge Base schema. No value specified for Inquisitor Import for z/OS UNIX.

FILTERSCHEMA=
: Default is IQF7. Inquisitor filter schema.

COMMIT=
: Default is 1000. Number of records stored before issuing a COMMIT.

Chapter 11. The Match Engine

The Match Engine uses the Global Knowledge Base (GKB), and the Local Knowledge Base (LKB) to match load modules in the Inquisitor tables to products at the Version Release Modification (VRM) level. The Match process has four phases:

- Perfect Match - this phase looks for an exact match of a product in one library.
- Volser Match - this phase matches products over multiple libraries but on the same volume.
- Interlib Match - this phase matches libraries not matched previously and tries to match over multiple libraries in different volumes.
- Rules Match - this phase identifies a product for a library. It overrides any previously matched information.

Schemas and scorecards

The Match Engine program uses a scorecard algorithm to recognize groups of LMODs as program products. It uses temporary scorecard tables as working space and these tables are defined internally by the Match Engine program.

It also uses DB2 temporary work files as intermediate work areas and, by default, DB2 Global Temporary Tables are used. This means that temporary tables and indexes are created and dropped after each Match Engine session.

Note: In DB2 Version 8, ″Declare Global Temporary Table″ requires 8K table space to be defined in a TEMP database.

Running the Match Engine

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Match Engine. Members beginning with ZME (UME for z/OS UNIX), and suffixed with the schema name, contain JCL to run the Match Engine for a given schema.

Run times for these jobs depend on the number of modules to be matched. It is recommended to run these jobs during off-peak periods.

TPARAM parameters

SSID= DB2 subsystem name. Value assigned, as defined in job HSISCUST.

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

DATABASE=
Name of product database. Value assigned, as defined in job HSISCUST.

IQSCHEMA=
Inquisitor schemas. Value assigned, as defined in job HSISCUST.

GKBSCHEMA=
Default is GKB7. Global Knowledge Base schema.

LKBSCHEMA=
Default is LKB7. Local Knowledge Base schema.

COMMIT=
: Default is 1000. Number of records stored before issuing a COMMIT.

LOGPERCENTAGE=
: Default is 7. Amount of detail produced in Match Engine report.

MECLEAROPTIONS=
: Default is 2. This clears all matches from the Inquisitor. A value of 0 means you should not clear the Inquisitor. A value of 1 means you should clear unknowns only.

UDAMBUFSIZE=
: Default is 153600000. This is the amount of REAL memory that is allocated to the Match Engine in order to store the scorecard entries. The size of this parameter determines the performance of the Match Engine.

SPILLPFX=
: Default is DB2I81. This is a high level qualifier for temporary flat files. It is used when the memory defined in UDAMBUFSIZE is exceeded.

IXBUFFERPOOL=
: Default is BP1. Buffer pool for indexes created on declared temporary tables.

Chapter 12. Load to Repository

Load to Repository is the process of copying your matched Inquisitor data into the Repository, where all the Inventory data is stored. Once an Inquisitor has been matched by the Match Engine, it is ready to be loaded to the Repository. Load to Repository copies the Inventory information from the Inquisitor into the Repository, making it available for standard queries and for viewing alongside other systems' inventories.

The generation of the high-level qualifier listing for identified products is included as a job step in the Load to Repository job. This listing is written to the UM.HLQIDS sequential data set, and is referenced by the Usage Monitor. For further details, see "High-level qualifier listing for Usage Monitor" on page 125.

Continuous Inventory

It is possible to replace an already existing Inventory in the Repository, with data from an updated Inquisitor. You might have added a new version of a product to production, and because the product resides in the same library as the old one did, it is much easier to just do an Inquisitor of the updated library and then replace the old version in the Repository, without losing the Usage data from the previous version.

A check is done by the system to see if a library in the Inquisitor matches one currently in the Repository. If it is an exact match, a further check is performed to see if the load module names match. If the two matches agree, then rename the old modules with an extension of VRM, for example, HSICKBME_4.1.0, and copy in the new modules from the Inquisitor. This enables you to only update a single Inventory, rather than create a new one each time you run an Inquisitor. This forms the basis of having a Continuous Inventory where the INVNAME you specify is used to check if an Inventory already exists. The Inventory name specified here must be identical to the Inventory name you want to replace, otherwise a new Inventory name is created.

If you need to refresh your Repository with a complete Inquisitor data, then use parameter **REPLACEFULL=Y**. This parameter marks as deleted any products that no longer exist in the Inquisitor data, but an entry still exists in the Repository. This allows you to report on products that have been deleted.

Note: REPLACEFULL is only to be used when a complete Inquisitor and match has been done. If you are just adding extra Inquisitor data to the Repository, then make sure **REPLACEFULL** is equal to N. The default value is N.

Adding Inquisitor data to an existing Inventory

In Tivoli Asset Discovery for z/OS you can add Inquisitor data to an existing Inventory, as long as the Inquisitor data you want to add has not been loaded before. To do this you must first determine the name of the Inventory you wish to add to. Run this SQL statement:

```
SELECT * FROM REPschema.TINVCTL.
```

This lists the Inventories currently in the selected repository. Make a note of the Inventory name.

Edit the ZLR*schema* (ULR*schema* for z/OS UNIX) job and add the keyword INVNAME=, and the name of the Inventory you wish to add to the TPARAM DD statement. The name has to be exactly the same as an existing Inventory name.

Modify the other statements as required. Submit the job. This adds new Inquisitor data to an existing Inventory in the Repository.

Running Load to Repository

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Load to Repository function. Members beginning with ZLR (ULR for z/OS UNIX), and suffixed with the schema name, contain JCL to run the Load to Repository for a given schema.

Run times for these jobs depend on the number of modules to be loaded into the DB2 Repository tables. It is recommended to run these jobs during off-peak periods.

TPARAM parameters

SSID= DB2 subsystem name. Value assigned, as defined in job HSISCUST.

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

DATABASE=
Name of database. Value assigned, as defined in job HSISCUST.

INVNAME=
Default value is the same as the IQ schema name. This is the name of the Inventory.

Note: The value specified in the z/OS UNIX TPARAM should be identical to the one specified here.

IQSCHEMA=
Inquisitor schemas. Value assigned, as defined in job HSISCUST.

GKBSCHEMA=
Default is GKB7. Global Knowledge Base schema.

REPSCHEMA=
Repository schema.

COMMIT=
Default is 1000. Number of records stored before issuing a COMMIT.

REPLACEFULL
Default is N. Y means that you wish to replace the current Inventory with a complete Inquisitor.

MIGUNIDENT=
Default is N. Y means that you wish to load all modules, including unidentified modules.

Chapter 13. Usage Import

Usage Import is the process of importing Usage data into the Repository. This process matches Usage data to Products that have been defined in the Repository.

An extra job step is now included as part of the Usage Import run. This is the Repository Merge whereby the summary tables in the Repository are rebuilt.

Running Usage Import

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Usage Import function. Member HSISUIMP contains JCL to run the job.

Run time for this job depends on the amount of Usage data to be loaded into the DB2 Repository tables. It is recommended to run this job during off-peak periods.

TPARAM parameters

SSID= DB2 subsystem name. Value assigned, as defined in job HSISCUST.

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

DATABASE=
Name of database. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository schema.

SYSPLEX=
Default is N. Change to Y if you are in a sysplex environment.

COMMIT=
Default is 1000. Number of records stored before issuing a COMMIT.

Chapter 14. Reporting

Reporting using Tivoli Common Reporting

The Tivoli Asset Discovery for z/OS Report set in Tivoli Common Reporting has the following branches:

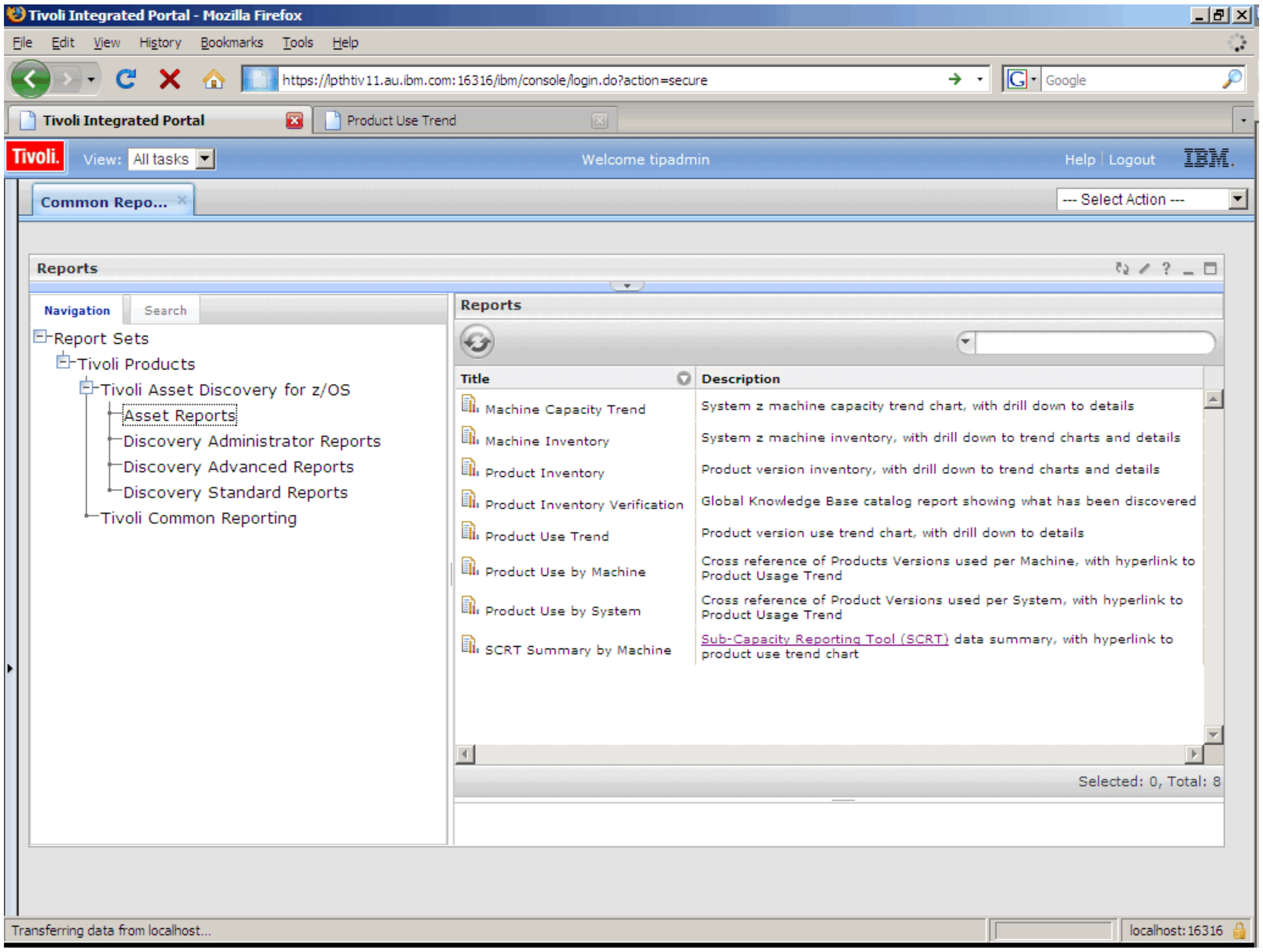
Asset Reports
These reports focus on Hardware and Software Assets. Note, that software product discovery is performed at a release level for accuracy. This data is then aggregated up to the product version level, at which products are licensed and information is displayed in the Asset Reports.

IBM Tivoli IT Asset Management also queries the product version aggregated data.

Discovery Standard Reports
Standard reports about discovered products at a release level.

Discovery Advanced Reports
Advanced reports about discovered products at a release level.

Discovery Administrator Reports
Administrator reports about discovered products at a release level

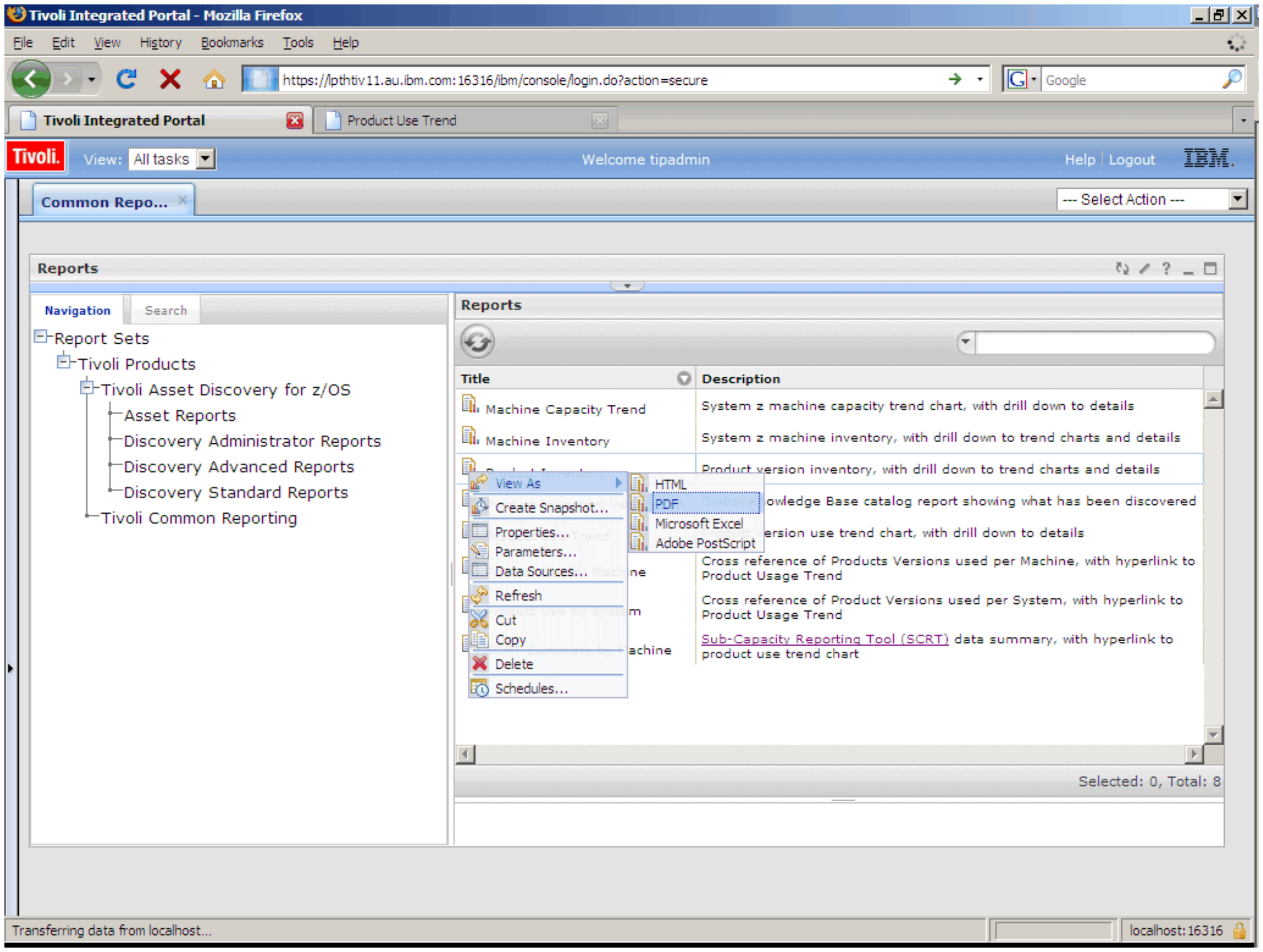

Tivoli Integrated Portal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://lpthtiv11.au.ibm.com:16316/ibm/console/login.do?action=secure
Google
Tivoli Integrated Portal
Product Use Trend
Tivoli.
View: All tasks
Welcome tipadmin
Help | Logout
IBM.
Common Repo...
--- Select Action ---
Reports
Navigation
Search
Report Sets
Tivoli Products
Tivoli Asset Discovery for z/OS
Asset Reports
Discovery Administrator Reports
Discovery Advanced Reports
Discovery Standard Reports
Tivoli Common Reporting
Reports
Selected: 0, Total: 8
Transferring data from localhost...
localhost:16316

Title	Description
Machine Capacity Trend	System z machine capacity trend chart, with drill down to details
Machine Inventory	System z machine inventory, with drill down to trend charts and details
Product Inventory	Product version inventory, with drill down to trend charts and details
Product Inventory Verification	Global Knowledge Base catalog report showing what has been discovered
Product Use Trend	Product version use trend chart, with drill down to details
Product Use by Machine	Cross reference of Products Versions used per Machine, with hyperlink to Product Usage Trend
Product Use by System	Cross reference of Product Versions used per System, with hyperlink to Product Usage Trend
SCRT Summary by Machine	Sub-Capacity Reporting Tool (SCRT) data summary, with hyperlink to product use trend chart

Figure 8. Tivoli Common Reporting navigation tree

The reports are generated by clicking on the icon next to the report title. By default, the reports are shown in your HTML Browser. These reports can be saved by using your Browser **Menu File** + **Save**.

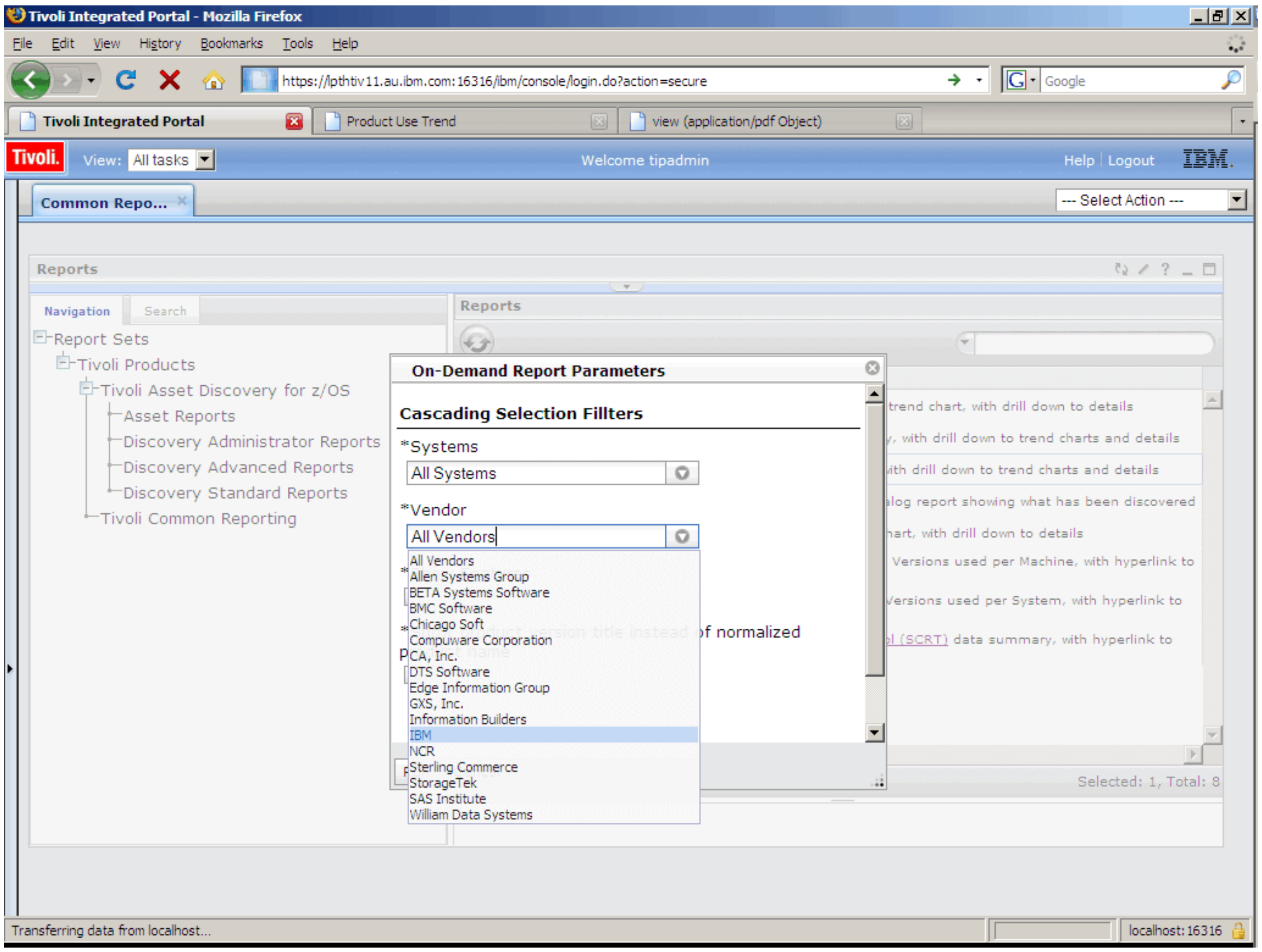
All reports can also be generated in PDF and Excel formats by right-clicking the report and selecting **View As**.



Title	Description
Machine Capacity Trend	System z machine capacity trend chart, with drill down to details
Machine Inventory	System z machine inventory, with drill down to trend charts and details
	Product version inventory, with drill down to trend charts and details
	owledge Base catalog report showing what has been discovered
	ersion use trend chart, with drill down to details
ne	Cross reference of Products Versions used per Machine, with hyperlink to Product Usage Trend
m	Cross reference of Product Versions used per System, with hyperlink to Product Usage Trend
achine	Sub-Capacity Reporting Tool (SCRT) data summary, with hyperlink to product use trend chart

Figure 9. Tivoli Common Reporting Right-click View As

Most reports have parameter prompts, including drop down selection lists that are dynamically generated from queries of the database.


Tivoli Integrated Portal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://lpthtiv11.au.ibm.com:16316/ibm/console/login.do?action=secure
Google
Tivoli Integrated Portal
Product Use Trend
view (application/pdf Object)
Tivoli.
View: All tasks
Welcome tipadmin
Help | Logout
IBM.
Common Repo...
--- Select Action ---
Reports
Navigation
Search
Report Sets
Tivoli Products
Tivoli Asset Discovery for z/OS
Asset Reports
Discovery Administrator Reports
Discovery Advanced Reports
Discovery Standard Reports
Tivoli Common Reporting
Reports
On-Demand Report Parameters
Cascading Selection Fillters
Selected: 1, Total: 8
Transferring data from localhost...
localhost:16316

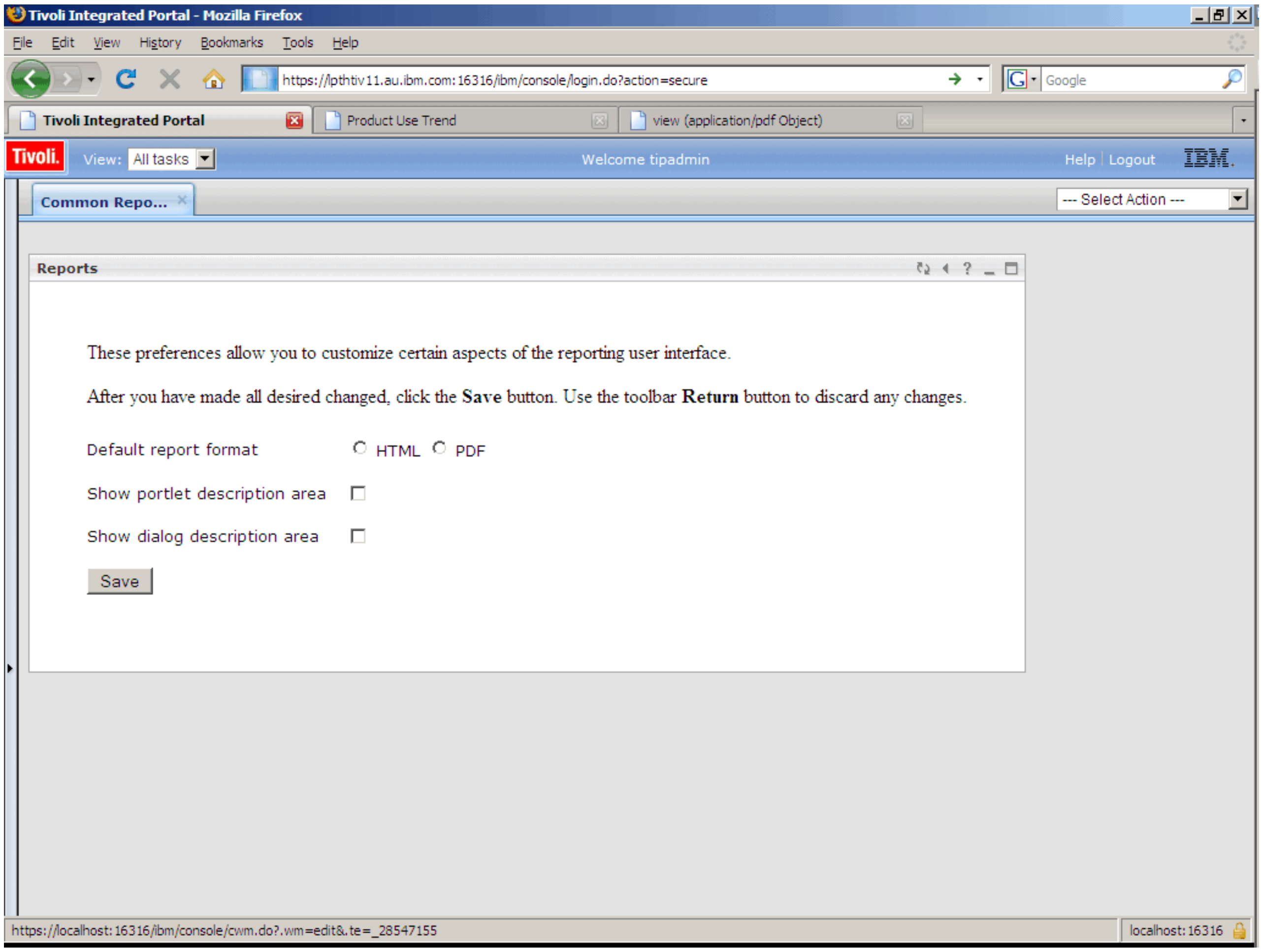
*Systems: All Systems

*Vendor: All Vendors

All Vendors
Allen Systems Group
BETA Systems Software
BMC Software
Chicago Soft
Compuware Corporation
CA, Inc.
DTS Software
Edge Information Group
GXS, Inc.
Information Builders
IBM
NCR
Sterling Commerce
StorageTek
SAS Institute
William Data Systems

Figure 10. Tivoli Common Reporting Example Parameter Prompt

Click on the / icon on the top right to change the default settings. For clearer parameter prompting, it is recommended to un-check the Show dialog description area setting. The settings are saved using Tivoli Common Reporting user ID.


Tivoli Integrated Portal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://lpthtiv11.au.ibm.com:16316/ibm/console/login.do?action=secure
Tivoli Integrated Portal
Product Use Trend
view (application/pdf Object)
Tivoli.
View: All tasks
Welcome tipadmin
Help Logout
IBM.
Common Repo...
--- Select Action ---
Reports

These preferences allow you to customize certain aspects of the reporting user interface.

After you have made all desired changed, click the **Save** button. Use the toolbar **Return** button to discard any changes.

Default report format ○ HTML ○ PDF

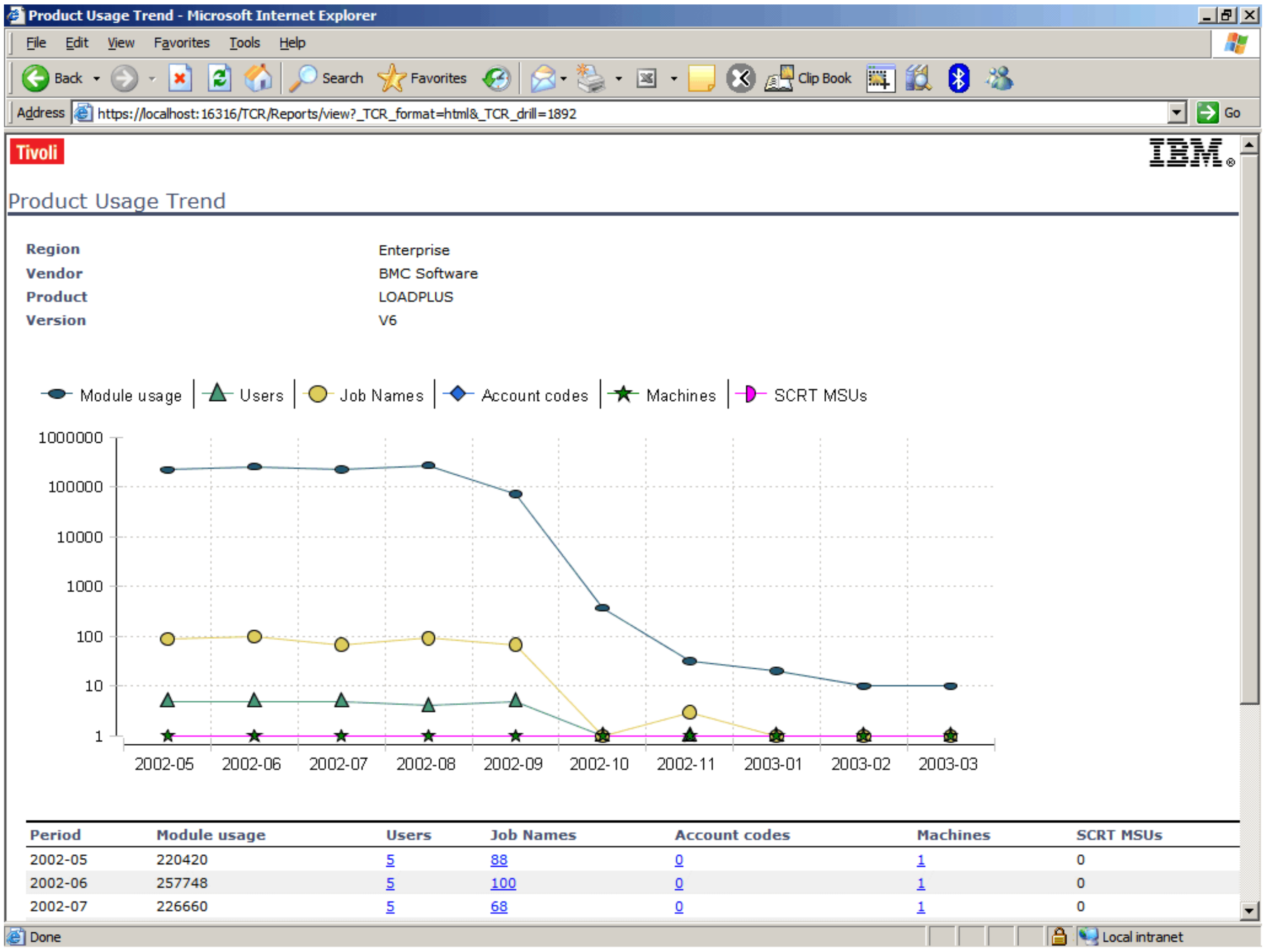
Show portlet description area ☐

Show dialog description area ☐

Save
https://localhost:16316/ibm/console/cwm.do?.wm=edit&.te=_28547155
localhost:16316

Figure 11. Tivoli Common Reporting Edit User Preferences

Most reports have hyper links than can be clicked to drill down for more details. This is particularly valuable for inspecting trend charts.


Product Usage Trend - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Clip Book
Address https://localhost:16316/TCR/Reports/view?_TCR_format=html&_TCR_drill=1892
Go
Tivoli
IBM
Module usage
Users
Job Names
Account codes
Machines
SCRT MSUs
1000000
100000
10000
1000
100
10
1
2002-05
2002-06
2002-07
2002-08
2002-09
2002-10
2002-11
2003-01
2003-02
2003-03
Done
Local intranet

Product Usage Trend

Region	Enterprise
Vendor	BMC Software
Product	LOADPLUS
Version	V6

Period	Module usage	Users	Job Names	Account codes	Machines	SCRT MSUs
2002-05	220420	5	88	0	1	0
2002-06	257748	5	100	0	1	0
2002-07	226660	5	68	0	1	0

Figure 12. Tivoli Common Reporting Example Trend Chart with Hyperlinks

Asset Reports

Table 11. Hardware and Software Assets reports

Report Title	Report Description
Product Inventory	Product version inventory, with drill down to trend charts and details
Product Inventory Verification	Global Knowledge Base catalog showing what has been discovered
Product Use Trend	Product version use trend chart, with drill down to details
Product Use by Machine	Cross Reference of Product versions used per Machine, with hyperlink to Product Use Trend
Product Use by System	Cross Reference of Product versions used per System, with hyperlink to Product Use Trend
SCRT Summary by Machine	Sub-Capacity Reporting Tool (SCRT) data summary, with hyperlink to Product Use Trend
Machine Inventory	System z machine inventory, with drill down to trend charts and details
Machine Capacity Trend	System z machine capacity trend chart, with drill down to trend charts and details

Discovery Standard Reports

Table 12. Standard reports about discovered products at a release level

Report Title	Report Description
Product Summary	Summary of discovered product releases
Product Detail	Details of discovered product releases
Products by Machine	Discovered product releases per machine
System Enterprise Browse	Browse Regions, with drill down for more details

Discovery Advanced Reports

Table 13. Advanced reports about discovered products at a release level

Report Title	Report Description
Search for Product Usage	Search product release use data
Low Product Usage	Product releases with low usage
APF Authorized Libraries	APF library summary
Deleted Libraries	Reports libraries that have been deleted
Deleted Products	Reports products that have been deleted
Unused Modules	Reports libraries with unused modules
Unused Products	Reports unused product releases
Unused Products and Libraries	Reports unused product releases and libraries

Discovery Administrator Reports

Table 14. Administrator reports for Tivoli Asset Discovery for z/OS

Report Title	Report Description
Installation Verification	Verify the connections to the databases are okay
IQ Database Report	Browse IQ libraries and modules
IQ Filters	List of IQ filters
Knowledge Base Reports	Knowledge Base catalog
Not Identified Modules	Modules with usage for not identified products, for example customer applications
Products with unknown release	Modules with unknown product release

Batch reporting

Ad hoc batch reports are provided as part of this product, and are run from a supplied batch job. SQL queries to produce these reports are supplied as members in the PARMLIB data set.

As part of online reporting, you can also run the same SQL queries using SPUFI or QMF™.

The reports are listed here, some of which might be familiar if you are a user of Tivoli License Compliance Manager for z/OS:

Table 15. Tivoli Asset Discovery for z/OS ad hoc reports

Member	Description
HSISIPID	INSTALLED-PRODUCT-IDENTIFICATION-DETAIL
HSISIPV	INSTALLED-PRODUCT-SUMMARY
HSISLLID	LOAD-LIBRARY-IDENTIFICATION-DETAIL
HSISLLS	LOAD-LIBRARY-SUMMARY
HSISLMD	LOAD-MODULE-DETAIL
HSISMUSL	MODULE-USE-SUMMARY-BY-LIBRARY-AND-PRODUCT
HSISMUSP	MODULE-USE-SUMMARY-BY-PRODUCT-AND-LIBRARY
HSISPDLP	PRODUCT-USE-DETAIL-BY-LIBRARY-AND-PRODUCT
HSISPDPL	PRODUCT-USE-DETAIL-BY-PRODUCT-AND-LIBRARY
HSISPJD	PRODUCT-USE-DETAIL-BY-JOBNAME-PRODUCT-AND-LIBRARY
HSISPJP	PRODUCT-USE-SUMMARY-BY-JOBNAME-AND-PRODUCT
HSISPJPL	PRODUCT-USE-SUMMARY-BY-JOBNAME-PRODUCT-AND-LIBRARY
HSISPSLP	PRODUCT-USE-SUMMARY-BY-LIBRARY-AND-PRODUCT
HSISPSP	PROD-USE-SUMMARY-BY-PRODUCT
HSISPSPL	PRODUCT-USE-SUMMARY-BY-PRODUCT-AND-LIBRARY
HSISPUD	PRODUCT-USE-DETAIL-BY-USERID-PRODUCT-AND-LIBRARY
HSISPUP	PRODUCT-USE-SUMMARY-BY-USERID-AND-PRODUCT
HSISPUPL	PRODUCT-USE-SUMMARY-BY-USERID-PRODUCT-AND-LIBRARY
HSISPV	PRODUCTS-BY-VENDOR
HSISVS	VERSION-SUMMARY
HSISLINV	List all inventories in DB2 Repository

When running in batch, ensure that the DB2 REXX environment for the target DB2 system has been successfully set up before using the samples.

Running the sample HSISBATR batch reporting JCL

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run batch reporting. These reports are generated from the data held in the product DB2, using a combination of DB2, REXX, and SQL.

Before running this job, ensure it has been updated with the following information:

1. The DB2 target subsystem name where Tivoli Asset Discovery for z/OS is defined.
2. Remove the comments in the SYSIN DD for the report that you wish to run. Note that multiple reports can be executed in a single job.
3. Ensure that DB2 has been bound for REXX language support.

In order to avoid producing large amounts of data, a ceiling of 1000 rows has been set as an output limit for all reports. This can be removed or modified by editing the SYSIN member corresponding to the report you would like to run.

Delete or modify the following SQL statement at the bottom of each member:

```
FETCH FIRST 1000 ROWS ONLY
```

The supplied samples are designed to demonstrate how to report on a subset of the data that has been collected by the Inquisitor and the Usage Monitor in order to reduce the amount of resources required by DB2.

For example, several samples have this SQL inserted:

```
AND TPRODUCT.FPRODUCTNAME LIKE '%IBM%'
AND TLIBRARY.FLIBNAME LIKE 'SYS%'
```

This restricts the output to only product names containing the string IBM, and data set names that begin with the string SYS. You may wish to review the SQL before running.

For performance reasons, when running SQL queries in DB2 that produce a lot of output, you have to increase the sizes of the table spaces of the work file database.

Chapter 15. Utilities

There are a number of utility jobs that are run in batch that enable you to administer the product, especially the Repository.

Knowledge Base Updates

The product's knowledge databases are updated monthly. In order to have the latest updates installed on your system, you need to download the updates from the IBM FTP server. The file that contains the updates contains z/OS, USS, and Inquisitor filter updates. Each needs to be loaded separately. The reload jobs drop the Knowledge Base tables and then reload the updates into empty tables.

Running the Knowledge Base Updates

The name of the file that contains the updates is TADZKB.XMI.

Download this file as binary. You then need to upload this file to the mainframe to a pre-allocated file with the attributes FB 80.

Once uploaded, you need to receive the file, RECEIVE INDATASET(tadzkb.xmit).

When prompted for additional information enter, DA(name of file)

The name of the file is important as it is used as input to the KB load jobs.

This restores the KB file, making it ready to be used to update the Knowledge bases.

Once you have restored the KB, you need to run the KB load jobs.

1. Before you submit any job, ensure that the input file is changed to the name of the file you have received it from.
2. For HSISDB03 in the JCLLIB:
   - Uncomment STEP3. This step drops the Global Knowledge Base tables.
   - Submit HSISDB03.
3. For HSISDB05 in the JCLLIB:
   - Uncomment STEP3. This step drops the Global Knowledge Base for z/OS UNIX tables.
   - Submit HSISDB05.
4. For HSISDB11 in the JCLLIB (optional):
   - Comment STEP1. This prevents creation of table space
   - Uncomment STEP3. This drops the IQ Filter tables
   - submit HSISDB11
5. If you want to be notified about Knowledge Base updates by email, contact IBM Support to add your name to the mailing list.

Automation Server for z/OS

The Automation Server is a feature provided with Tivoli Asset Discovery for z/OS, used to automate the processing of data sets and JCL required by the product.

The Automation Server runs in its own address space as a started task. Ensure that the user ID used by the Automation Server has an OMVS segment and UID, or use of a default UID.

Input control statements define the actions to be performed by the Automation Server. Two types of actions are supported:
1. FTP, which invokes the FTP utility to perform a file transfer.
2. JOB, which submits a batch job.

Each action definition is associated one or more data set name masks.

A specified range of times, within a 24 hour period, decide how often a catalog search is performed to determine if any data sets with names matching the mask need to be processed.

When a new data set is found, the associated action is performed.

The Automation Server has a VSAM control data set, which is updated with the status of each data set action. This is to allow for the use of FTP retries and to ensure that each action is performed once only for each data set.

To aid users in setting up the Automation Server, job HSISCOMB is supplied in PARMLIB, as an example. This is a combination of the Inquisitor Import, Match Engine, and the Load to Repository in a single job stream. This job is submitted by the Automation Server once the Inquisitor file has been closed. It is only set up for the first IQ schema that is defined in the HSISCUST job. In order to use it with other IQ schemas, make a copy of this job, and rename the IQ schema name to the one you want to use.

Automation Server JCL

The Automation Server started task JCL is supplied in the SHSIPROC data set.

The member HSIJAUTO is used to start the Automation Server. The JCL procedure is to be copied to a library in the JES PROCLIB concatenation.

The parameters listed here are used in the started task JCL for the Automation Server:

HSI	High Level Qualifiers for the Installation Target Libraries months
HSIINST	High Level Qualifiers for the HSIINST..PARMLIB data set created by the HSISCUST job.
ACDS	Data set Name of the ACDS (Automation Control Data Set).

Files used by the Automation Server

The Automation Server uses the files listed here:

STEPLIB
: Load library containing the product software. Not required if Tivoli Asset Discovery for z/OS is installed into the system link list.

HSIACNTL
: Partitioned data set containing fixed length 80-byte records. Member HSIAPARM contains the Automation Server control statements, which specify the actions to be performed. For each action in the HSIAPARM member, there is a corresponding member containing the template data for that action. The template data is JCL or an FTP command stream, containing symbolic references to be resolved by the Automation Server whenever the action is performed.

HSIACDS
: A VSAM KSDS used by the Automation Server. Initialization statements, error messages, and activity logging messages, are written to this file.

HSIAMSG
: Specifies the message report file for the Automation Server. Initialization statements, error messages, and activity logging messages, are written to this file.

SYSPRINT
: Specifies the message report file for PL/I.

SYSOUT
: Specifies the message report file for Language Environment.

OUTPUT
: Specifies the message report file for the FTP program. Its contents are determined by the FTP program installed in the system.

INPUT
: Specifies a fixed length 120 byte record file containing FTP commands read by the FTP program. The FTP commands are written to this file before FTP is attached by the Automation Server whenever an FTP action is to be performed.

INTRDR
: Specifies a fixed length 80 byte record file to be directed to the system's internal reader. The Automation Server writes a job stream to this file whenever a JOB action is to be performed.

Creating the Automation Server control data set

The HSISCUST post-installation customization job creates member HSIASALC in the JCLLIB in order to create the Automation Server Control data set.

Member HSIASALC contains IDCAMS JCL and control statements to create the control data set.

The Automation Server control data set is a VSAM KSDS, which the Automation Server allocated with the HSIACDS ddname.

Note: The Automation Server control data set must have sufficient space allocated to it in order to handle the workload required by the installation. One 96 byte record, including the 52 byte key, is required for each data set processed by the Automation Server.

Excluding data sets from Automation Server processing

In the HSIAPARM member you define actions to be performed, and supply data set name masks specifying the data sets to be processed. Data sets with these name patterns might already exist, and have already been processed by other means before the Automation Server is implemented. In this case, you need to inform the Automation Server that these data sets are not to be processed. Data sets that you want excluded from the Automation Server processing must have a record in the Automation Server control data set indicating that the data set has already been processed. The name of the data set you want excluded must satisfy a selection mask pattern.

This can be achieved using the Automation Server data set name scouting program. This reads the HISAPARM member and searches the catalog for every specified data set name mask. If a data set is found, a record is written, then sorted into key order, and copied into the VSAM control data set.

Every record loaded into the control data set in this way indicates a specific action, and has that action flagged as complete.

You can edit the sequential, before the data is copied into the control data set, to manually delete any records for data sets you still want processed by the Automation Server.

The HSISCUST post-installation customization job creates member HSIASSCT in the JCLLIB to run the scouting program.

Starting the Automation Server

When installed as a started task, the Automation Server must be started by an operator START command. Ensure that the user ID assigned to the Automation Server has RACF CONTROL access to the VSAM data set.

Stopping the Automation Server

Whether running as a started task or as a batch job, the Automation Server is stopped by an MVS STOP command.

Action processing

The Automation Server can perform two different actions, FTP and JOB. In both cases, when an action is to be performed, the nominated template member is used as input. If the template member cannot be found, an error message is written to the file identified by ddname HSIAMSG. The Automation Server checks every 10 minutes to see if there is any work to perform.

Each 80 byte record is read from the template member and searched for the ampersand character, the presence of which triggers a call to the system’s symbol substitution routine, ASASYMBM.

The name of the data set which triggers this action, is found by the catalog search and fed into the output stream. After any symbol substitution, the records are written to a sequential file, which is closed when the end of the template data is reached.

For action FTP, the sequential input file is pointed to by the INPUT DD statement. The FTP is attached as a subtask. FTP scans the INPUT file and processes the FTP requests therein. FTP writes its report messages to the OUTPUT file. For the Automation Server, the INPUT file must be a temporary data set and VIO is ideal for this purpose.

When the FTP subtask completes, the Automation Server examines the completion code. If the FTP program ends normally with a zero return code, then the Automation Server deems the action to have been successful and updates the action status in the HSIACDS file. This way, the action is not repeated for this data set.

If the FTP program abends, or does not end normally with a return code of zero, the Automation Server deems the action to have failed. A failed transfer is retried at a later time. Retries are subject to any specified time-of-day window constraints. You should examine the OUTPUT FTP report file to determine the exact cause of any transfer failure.

For action JOB, the JCL generated is pointed to by the INTRDR DD statement. The INTRDR file is directed to the system's internal reader, and the jobs submitted by the Automation Server become available fro JCL conversion as soon as the INTRDR file is closed, or another JOB card image is encountered by the reader.

Note: A job stream can contain more than one job.

The Automation Server deems all job submissions successful, so there are no retries performed for JOB actions. There is no feedback from the job's execution status to the Automation Server.

It is assumed that your installations have appropriate procedures in place to handle any failures in jobs submitted by the Automation Server.

In principle, an action to invoke FTP to transfer a file can be implemented as a JOB action, with the corresponding template containing an FTP batch job stream. However, such an implementation removes the ability of the Automation Server to track the success of the transfer, thereby precluding the Automation Server initiated retries.

Note: The Automation Server does not check the validity of JOB or FTP streams, but copies the records from the template member to the file appropriate for the action type, performing symbol substitution where necessary.

Automation Server control data set maintenance

A record is kept for every data set processed by the Automation Server in the Automation Server control data set, ASCDS. The primary purpose of this record is to prevent the repeated processing of a data set for the same data set name mask. As records accrue, the size of the data in the ASCDS continues to grow.

If a processed data set is deleted, or a data set name mask is removed from the set of masks processed by the Automation Server, then there is no reason to keep a record of that data set in the ASCDS. The Automation Server performs a cleanup cycle for the ASCDS on a daily basis.

The cleanup cycle consists of reading the ASCDS sequentially, and deleting records for data sets which have not been found by any catalog search. This is based on the relevant data set name mask in the current calendar month, or in the prior calendar month.

As with most VSAM data sets with ongoing record insertion and deletion activity, it is advisable to regularly reorganize the ASCDS periodically.

A backup regime for the ASCDS must be robust and have the frequency which corresponds with the adverse consequences of reprocessing any already processed data set.

Automation Server request control statements

The Automation Server action requests are specified in the HSIAPARM member of the SHSIPARM file.

Syntax rules are as follows:
- Records beginning with an asterisk are comments.
- Blank records are comments.
- Verbs and operands must begin before column 72.
- No blanks are allowed between a verb or operand and its opening parenthesis.
- A subparameter in parentheses can have leading or trailing blanks.
- Continuations on to subsequent records are not allowed.

Action statement

Function

Each statement requests that an action is performed for a data set whenever it matches an associated data set name mask. An action is only performed once for each match, but the presence of a data set triggers the action for each mask it matches.

An optional time-of-day window is specified for each action. Such a window indicates that searches of the catalog for data sets matching associated masks are only to occur during the nominated time-of-day window. The time-of-day window is used to restrict job submission and file transfers to times which are appropriate to administer, such as overnight, or outside peak processing times.

Data set name mask statement

Function

Each data set name mask statement associates the specified data set name mask with the preceding action statement. It is, therefore, invalid for the HSIAPARM member to begin with a data set name mask statement. When a data set with a name matching the specified mask is first located, the action specified in the preceding action statement is triggered.

The data set name mask of NULLFILE is a special case. When a data set name mask with this exact value is processed by the Automation Server, no catalog search is performed, but the associated action is triggered as if a new cataloged data set matching the mask has been located. The AS symbols for the complete data set name, and the first data set name qualifier, both have values of the 8 byte string NULLFILE. Use the data set name mask of NULLFILE to trigger daily actions which do not depend on the creation of any particular data set.

Syntax

Action statement and DSN syntax

```
►►──action(──template──)──┬────────────────────┬──────────────────────►
                          └─TIME(hhmm-hhmm)─┘

►─DSN(data-set-name-mask──)──────────────────────────────────────────►◄
```

action FTP or JOB

template
name of a member in the SHSIPARM file.

TIME This operand is optional, and the default is TIME(0000-2400), which specifies no time-of-day constraint. When specified, *hhmm-hhmm* must be 9 contiguous characters made up of 4 decimal digits, a dash, and 4 decimal digits. Each *hhmm* value specifies a time-of-day in 24-hour hours and minutes. The minimum value is 0000 and the maximum value is 2400, while the last 2 digits cannot exceed 59. The first *hhmm* specifies the time-of-day window start, and the second *hhmm* specifies the time-of-day window close. The second hhmm value must be greater than the first hhmm value.

hhmm-hhmm
specifies a time-of-day range.

DSN Data set name.

data-set-name-mask
Where *data-set-name-mask* does not exceed 44 characters in length, and specifies a data set name mask pattern acceptable to the Catalog Search Interface. The generic match mask for a single character is the percent sign. The generic match mask variable number of characters is the asterisk, and a double asterisk can be used to match a variable number of data set name qualifiers.

Control statement examples

Example 1:
Files created by the Usage Monitor undergo two independent processes, both within the 8:00 p.m. to 11:30 p.m. window. They are processed by a job based on the JCL contained in member HSISJOB, and then transferred to a z/OS system using the FTP commands in member HSISFTP1. All members are pointed to by the HSIACNTL ddname.

```
* TRANSFER USAGE MONITOR FILES TO Z/OS SYSTEM
   JOB(HSISJOB1)  TIME(2000-2330)
   DSN(USER.OMU*.D*.T*)
   FTP(HSISFTP1)  TIME(2000-2330)
   DSN(USER.OMU*.D*.T*)
```

Example 2:
Files created by the Usage Monitor are to be imported to the appropriate database.

```
* PERFORM USAGE MONITOR IMPORT
JOB(HSISUIMP)
DSN(USER.UMON.*.*)
```

In this example HSISUIMP contains the necessary JCL to run Usage Import on a z/OS system.

Note: The JCL can route the job to any connected NJE node, or specify affinity to any system sharing the SPOOL. Therefore, the job does not have to run on the z/OS system where the Automation Server is running. The template name, HSISUIMP in this example, need not match the job name submitted by the Automation Server action.

Automation Server symbol processing

Whenever an action is performed, the contents of the template member are written to an appropriate output file. Each 80 byte record is written unchanged, unless symbol substitution is required. If an ampersand character is present in a record from the template member, the system symbol substitution routine ASASYMBM is invoked to process the record before it is written. More than one symbol is allowed in a record. If an ampersand character does not denote the start of a recognized symbol, then that part of the data remains unchanged. Symbols available for use in template members include all z/OS system static symbols and symbols defined locally by the Automation Server. Automation Server local symbols are derived from the data set name of the catalog entry found which triggered the particular instance of the action.

Automation Server local symbols are provided in the list shown here:

&DATASETNAME.	The entire data set name.
&QUAL1.	The first qualifier of the data set name.
&QUAL2.	The second qualifier of the data set name.
&QUAL3.	The third qualifier of the data set name.
&QUAL4.	The fourth qualifier of the data set name.
&QUAL5.	The fifth qualifier of the data set name.
&QUAL6.	The sixth qualifier of the data set name.
&QUAL7.	The seventh qualifier of the data set name.
&QUAL8.	The eighth qualifier of the data set name.
&QUAL9.	The ninth qualifier of the data set name.

Automation Server symbol processing example

Example 1:

The data set triggering a JOB action is MACNIVE.IQ.ZIP. As a result, JCL DD statements referencing the data set in a template member can be represented like this:

```
//*-------------------------------------------------------------***
//* Sample JCL demonstrating the use of Automation Server local***
//* symbols derived from the data set name.                    ***
//*-------------------------------------------------------------***
//BR14      EXEC  PGM=IEFBR14
//DD1       DD    DSN=&&DATASETNAME.,DISP=SHR
//DD2       DD    DSN=&&QUAL1..&&QUAL2..&&QUAL3.,DISP=SHR
```

both of which would be resolved by symbol substitution to:

```
DSN=MACNIVE.IQ.ZIP,DISP=SHR
```

and this would be the DSN= JCL statement output to the internal reader.

Note: As symbol substitution is performed before the job is submitted, z/OS system symbols that are not allowed in batch job JCL are allowed in the Automation Server templates.

Product Tagging utility

The Product Tagging utility, referred to as the ″Tagger″, provides a means of identifying software which has not been predefined in the Global Knowledge Base (GKB). It can also be used to supersede GKB entries, without changing the GKB contents.

Product Tagging is currently a manual process where the details of the product, such as name and vendor, are provided, together with the location of the product's programs. The Tagger scans the programs in a similar fashion to the Inquisitor, and records the results for later collection and storage into the Local Knowledge Base (LKB), where it is used for software identification.

The SYSIN file contains several control statements describing the program product to be tagged, such as its name, the vendor's name, the product identifier, and version information. There is also one or more TAGLIBS statements supplied, which provide the Tagger with the information needed to find the program libraries containing the product's executable software.

The Tagger uses the TAGLIBS specifications to locate the relevant program libraries which are dynamically allocated and analyzed. Data sufficient to perform software identification for each relevant program is extracted.

Information about all of the discovered programs pertaining to the nominated program product is compiled into a single object module, which is either written to all of the relevant program libraries containing the product's programs, or written to the single program library allocated to the HSIREDIR file.

Installations can choose to keep all tag data separate from executable program product software by using the optional HSIREDIR file. If this is done, ensure that all relevant HSIREDIR file data sets are included in the normal Inquisitor scan processing, even if those data sets contain no other programs.

The Tagger can only process one software product or optional feature of a product in a single execution run.

All program object and load module libraries, excluding distribution libraries, of a product or product optional feature, are to be processed in the same Tagger execution run.

You can choose to overwrite the default output member name of @HSIPTAG by specifying a TAGMEM statement. All Tagger output members are flagged with an SSI value of X'D7E3C1C7', which is 'PTAG' in EBCDIC.

If there is no preexisting member of the same name, a new program member is created by the Tagger to contain the tag data. If a member already exists, the new tag data is added to the pre-existing data pertaining to other products or optional features. Any data pertaining to the same software piece identified by {VENDOR + PRODUCT + OPTION + VERSION} is replaced. Internally, the data relating to each software piece resides in its own control section. Tag data members contain no executable code and are bound with the only-loadable attribute. They are also bound as reentrant with a residence mode of ANY to minimize the impact of being placed in a library which is loaded into the Link Pack Area.

To erase the effects of Tagger processing, simply delete the tag data members which can be identified by their SSI value. If you are using ISPF to achieve this, the SORT SSI member list command is useful.

The tag data members created by the Tagger are recognized by the Inquisitor (by their SSI value) during normal program library scanning activity. The Inquisitor extracts the tag data from the member contents and writes it to its output file. The Inquisitor Import process uses these program tags to maintain entries for the programs in the Local Knowledge Base, thus enabling accurate identification by the Match Engine for the tagged product level, no matter which library the product is deployed to, and which system the Inventory data is collected from.

The software piece processed in a Tagger execution run is considered to have a key of {VENDOR + PRODUCT + OPTION + VERSION}. This means that, if non-key data items such as the values specified in the PPNUM or LICENSED statements are incorrect, they can be corrected by fixing the input statement values and rerunning the Tagger job, thereby replacing all non-key tag data. It also means that if a key data item is incorrect, it cannot simply be erased by running a Tagger job with the correct data.

If a product has multiple program libraries, there is no requirement for them to be processed under a single generic data set name mask. It is acceptable to have multiple TAGLIBS statements, with each selecting a single specific data set. It is also acceptable to have some TAGLIBS statements triggering catalog searches, and others triggering VTOC searches in the same Tagger execution run.

When processing libraries which are not dedicated to a single program product, use member name masking to prevent the tagging of programs not related to the specified product. Some installations deploy multiple software products in a combined common library. If the products are tagged before they are combined, use different tag data member names to allow all the tag data to reside in the common library.

Running the Tagger

The HSISCUST post-installation customization job creates JCL in the JCLLIB to run the Tagger. Member HSISPTAG contains JCL to run the job.

Tagger control statements

Control statements are input using the SYSIN file. General syntax rules are listed here::

- Fixed length, variable length, and undefined record formats are processed.
- Short records are extended to 72 bytes of data with blanks if necessary.
- Only the first 72 bytes of data for each record are processed by the Tagger.
- Records beginning with an asterisk are treated as comments and do not alter continuation status.
- The first nonblanks of a statement must identify the statement type.
- One or more blanks must follow the statement type.

- A statement with no value or operand specified is invalid.
- For statement types other than TAGLIBS, the specified value is deemed to start with the first nonblank after the statement type name.
- Statements are input in any order. All statements are processed before any tagging activity commences.
- TAGLIBS is the only statement type which can be supplied more than once in an input file.
- TAGLIBS is the only statement type which can be continued over more than one record.
- TAGLIBS is the only statement type where symbol substitution can be performed.

Here is a table of value-oriented statement types detailing all Tagger statement types except TAGLIBS. It shows the statement type name, a description of the meaning of its value, whether it is a mandatory or optional statement, the default value of optional input, and the maximum acceptable byte count.

Table 16. Tagger Statement Types

Statement Type	**Value**	**Default Value**	**Required**	**Maximum length**
VENDOR	Product's vendor name	-	Yes	30 bytes
PRODUCT	Product's name	-	Yes	50 bytes
PPNUM	Program product number	blanks	No	16 bytes
OPTION	Optional feature name	BASE	No	30 bytes
VERSION	Software level	-	Yes	8 bytes
LICENSED	Separately licensed feature? (YES or NO)	NO	No	3 bytes
TAGMEM	Output member name	@HSIPTAG	No	8 bytes

Note: The symbol @ represents the X'7C'code point.

TAGLIBS is not a value-oriented statement type, but has operands which have values specified in parentheses much like Inquisitor control statements. TAGLIBS does not currently have any keyword operands.

Operands can be abbreviated to the minimum unambiguous length, which currently means in practice that selection mask operands can be abbreviated to a single character, and exclusion mask operands can be abbreviated to two characters.

If the Tagger finds an ampersand (&) in a TAGLIBS statement record it invokes the system symbol substitution routine ASASYMBM to perform symbol substitution.

The Tagger stops parsing a TAGLIBS record and expects the current statement to continue on the next record whenever a continuation character is encountered. Valid continuation characters are plus (+) and dash (-). Continuation characters also function as delimiters, so a continuation cannot occur within an operand name, or a value mask.

The syntax description of TAGLIBS follows.

Syntax

```
►►──TAGLIBS──DATASET──DSNAME──(──dsname-mask──list──)─────────────────────►

►──┬──────────────────────────────────┬──┬──────────────────────────────────┬──►
   ├─XDATASET─────────────────────────┤  └─VOLUME──(──volume-mask──list──)──┘
   └─XDSNAME──(──dsname-mask──list──)─┘

►──┬──────────────────────────────────┬──┬──────────────────────────────────┬──►
   └─XVOLUME──(──volume-mask──list──)─┘  ├─PROGRAM──────────────────────────┤
                                         └─PGM──(──member-mask──list──)─────┘

►──┬──────────────────────────────────┬───────────────────────────────────────►◄
   ├─XPROGRAM─────────────────────────┤
   └─XPGM──(──member-mask──list──)────┘
```

dsname-mask
: a string up to 44 bytes in length, representing one or more possible data set names.

volume-mask
: is a string up to 6 bytes in length, representing one or more possible volume serial numbers. A mask of six asterisks is deemed to represent the IPL volume, which is ascertained during execution.

member-mask
: is a string up to 8 bytes in length representing one or more possible member names of a PDS and/or PDSE.

Masks in a mask list within the same set of parentheses must be separated by at least one delimiter. Acceptable delimiter characters are a blank, a comma, and a continuation character.

Multiple masks are allowed in each mask list. At least one data set name selection mask must be specified on each TAGLIBS statement. If an operand is specified more than once, then all masks specified in all occurrences of the operand are processed.

Program libraries are located either by invoking the Catalog Search Interface, or by scanning the VTOC of each volume which passes the volume selection and exclusion criteria. VTOC scans are performed if VOLUME or XVOLUME is specified; otherwise the catalog is searched.

The Tagger allocates a potential program library before determining if it is a program library. This is to ensure that it is on a primary volume, and that its VTOC entry is accessible. As a result, you need to check that the data set name masks used in a catalog search do not trigger excessive data set recalls.

Catalog searches are limited to entry type ’A’ non-VSAM entries, so data set aliases and GDG members are not processed. The Tagger does not process a data set unless it has partitioned organization, an undefined record format, and a block size of at least 1024. A PDSE is only processed if it is a program library.

When using generic masks, use percent to signal that any single character is to be considered a match in that exact location of the compared character string, and use an asterisk to signal that any zero or more characters are a match, with one exception now described.

Data set name mask matching performed by the Catalog Search Interface is qualifier-aware. The Catalog Search Interface treats an asterisk as part or whole of a single data set name qualifier. To specify any number of data set name qualifiers, use the double-asterisk. Only the masks specified in a DATASET or DSNAME operand are ever passed to the Catalog Search Interface by the Tagger.

When using generic data set name selection masks, you need to adjust the mask, depending on whether the data sets are to be located by catalog or VTOC searches. Do not use double asterisks in any mask other than data set name selection masks.

Region Association

The utility enables you to create region names and to associate the various inventories in the Repository with a region. Regions can contain other regions. Creating regions allows you to group together many inventories under a single name. This simplifies reporting based on a region, rather than individual inventories. You can create region names based on a geographic location; city or state.

Running Region Association

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Region Association utility. Member HSISREGN contains JCL to run the job.

TPARAM parameters

SSID= DB2 subsystem name. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository schema.

DBNAME=
Name of database. Value assigned, as defined in job HSISCUST.

SYSIN commands

The SYSIN commands specified in the SYSIN DD.

The syntax rules are:
1. Comments begin with a '*' in column 1.
2. Blank lines are allowed and are ignored.
3. Each statement must be terminated by a period (.).

The statements are:

LIST List region or inventory information.

Syntax

```
►►──LIST──┬───────────────────────┬───────────────────────────────►◄
          ├─REGion──region────────┤
          └─INVentory──inventory──┘
```

region specifies the name of a region and can be "ALL", a region name, or a region name including masking characters. If a region name includes spaces, enclose the name in quotes (").

inventory
specifies the name of an inventory and can be "ALL", an inventory name, or an inventory name including masking characters.

Masking characters are:

"*"	for one or more characters of any value.
"%"	for exactly one character of any value.

The keywords REGION and INVENTORY can be abbreviated to REG or INV.

DEFINE
The Define command creates region names that can be used to group inventories or other regions.

Syntax

```
►►──DEFINE──REGion──region──────────────────────────────────────►◄
```

region specifies the name of a region. If a region name includes spaces, enclose the name in quotes (").

The keyword REGION can be abbreviated to REG.

ASSIGN
The Assign command assigns an inventory to a region, or a region to a region. Assigning a region to another region makes the assigned region a child of the assigning region.

Syntax

```
►►──ASSIGN──┬───────────────────────┬──TO──region2──────────────►◄
            ├─REGion──region1───────┤
            └─INVentory──inventory──┘
```

region 1
specifies the name of a region to be assigned.

region 2
specifies the name of a region that the region or inventory is to be assigned to.

inventory
specifies the name of an inventory.

If a region name includes spaces, enclose the name in quotes(″).

The keywords REGION and INVENTORY can be abbreviated to REG or INV.

REMOVE
The Remove command removes an inventory assignment to a region.

Syntax

```
►►──REMOVE──INVentory──inventory──FROM──region──────────────────────►◄
```

inventory
specifies the name of an inventory that might have masking characters.

region specifies the name of a region that the inventory is to be removed from.

Masking characters are:

″*″	for one or more characters of any value.
″%″	for exactly one character of any value.

If a region name includes spaces, enclose the name in quotes(″).

The keyword INVENTORY can be abbreviated to INV.

DELETE
The Delete command deletes a region name.

Syntax

```
►►──DELETE──REGion──region──────────────────────────────────────────►◄
```

region specifies the name of a region. If a region name includes spaces, enclose the name in quotes (″).

The keyword REGION can be abbreviated to REG.

Add Inventory Association

When an Inventory is run in shared DASD-mode, all systems that provide Usage data for the Inventory need to be manually associated with an Inventory.

Creating an association

In order to create an association ,you must first ascertain what Inventories have already been loaded into the Repository, and the corresponding Inventory ID. Use the sample member HSISBATR from the customized JCLLIB data set. Uncomment the SYSIN DD statement which uses member HSISLINV from the customized PARMLIB data set as input. You can expect output similar to the output shown in this example:

```
REPORT: List all Tivoli Asset Discovery for z/OS inventories in DB2 Repository 2008-11-18 20.51.08

FINVID FINVNAME FVERSIONGKBID FIQDATE                    FIQDATEFIRST

______ ________ _____________ __________________________ __________________________
     1 PLEXD    36.15         2008-09-16-17.22.15.000000 2008-09-16-17.22.15.000000
     2 PLEXE    36.15         2008-09-16-17.00.51.000000 2008-09-16-17.00.51.000000
     3 PLEXR    36.15         2008-09-16-17.03.56.000000 2008-09-16-17.03.56.000000

FSYSPLEXID FSYSFMID FSMFID FPREVINVID FIQSYSPLEXUSE FVENDORCNT FPRODUCTCNT FLIBCNT FINVCONT FINVTYPE FIQNAME

__________ ________ ______ __________ _____________ __________ ___________ _______ ________ ________ ______
SYSPLEXD   HBB7730  ISD1            0             0          7         435    7523 Y               1 IQ33
SYSPLEXE   HBB7730  FAE1            0             0          6         235    1701 Y               1 IQ34
SYSPLEXR   HBB7730  BMR1            0             0          5         300    1281 Y               1 IQ35
```

Complete these two steps to associate an SMFID with an Inventory:

1. Identify the FINVID associated with the Inventory name (FINVNAME) that you want to update.
2. Update and submit member HSISINVA in the customized JCLLIB data set with your desired association.

For example, using the output produced by the SQL in member HSISLINV, it associates the SMFID FAE1 to the Inventory ID of two, in this case Inventory Name PLEXE.

```
INSERT INTO REPschema.TUIMPORTCTRL VALUES(2,0,0,0,0,0,0,0, ' ', 'FAE1');
```

With this utility, you can associate a single Inventory to eight SMFIDs. Usage Import checks the first Inventory to see if it has matching data. If so, then the Usage data is added to that Inventory. If not, then the next Inventory is checked.

Running Add Inventory Association

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Add Inventory Association. Member HSISINVA contains JCL to run the job.

SYSIN Parameter

```
DELETE FROM &REPSCHMA..TUIMPORTCTRL WHERE FLPARNAME = 'AAV1' ;
```

Delete Inventory Association

When an Inventory is no longer associated with an SMF ID, the association should be deleted.

Running Delete Inventory Association

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Delete Inventory Association. Member HSISASSD contains JCL to run the job.

SYSIN Parameter

```
DELETE FROM &REPSCHMA..TUIMPORTCTRL WHERE FLPARNAME = 'AAV1' ;
```

Usage Summary

Usage Summary is the process of summarizing Usage data in the Repository. This process deletes detailed Usage records and creates monthly summary records, by reducing the number of DB2 rows used to represent your old data.

To minimize space utilization and improve SQL query performance, it is recommended that you keep detailed module Usage data for the last three months and summarize all detailed module Usage data older than three months. We also recommend that you delete summarized module Usage data older than 18 months. Please see job HSISUDEL (Usage Deletion) for more details.

If you have not run the Usage Summary job for some time, then select a period of a few months at a time, in order to keep the run times down to a reasonable time.

Running Usage Summary

The HSISCUST post-installation customization job creates JCL in the JCLLIB to run Usage Summary. Member HSISUSUM contains JCL to run the job.

TPARAM parameters

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

DATABASE=
Name of product database. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository schema.

SUMBY=
The parameter is used to summarize the Usage data by either consumer (SUMBY=1), by module (SUMBY=2), or both (SUMBY= 3).

SUMBY=1
Summarize by Consumers. Consumer information (JOB NAME & USER ID) is ignored, with usage events being attributed to the JOB TYPE (BATCH, TSO, DB2...)

SUMBY=2
Summarize by Module. Load module (=program) names are ignored, with the value *MODULE* as a placeholder.

SUMBY=3
Summarize by both Consumers and Modules. The rules for SUMBY=1 and SUMBY=2 apply.

Note: You can choose to summarize by module only, and later on summarize the same data, except this time you choose consumer. If you choose to summarize by consumer, then all usage queries would no longer display the consumer name, but the type of process that the user was using. For example, you would have consumer names like TSO, DB2, and Batch . For module usage, queries would only have a module name of *MODULE* .

FIRSTDATE=
Start of the first date range. This is in the form YYYYMM. Only complete months are chosen.

LASTDATE=
End of the last date range. This is in the form YYYYMM.

Note: The date range of summarization is inclusive of the month specified in the FIRSTDATE and LASTDATE parameters.

MINUSAGETHRESHOLD
Default is 1000. Sets a value for Usage Summary to ignore summarization of usage records. If this parameter is set to 1000, then any product with a usage count of 1000 or less for any given month, does not have its detail data summarized. This allows you to view the detail data for low usage products.

COMMIT=
Default is 1000. Number of records stored before issuing of COMMIT.

Usage Deletion

Usage Deletion is the process of deleting Usage data in the Repository. This process deletes detailed, summarized and aggregated Usage data for the period specified for all Inventories in the Repository.

To minimize space utilization and improve SQL query performance, it is recommended that you keep no more than 3 months of detailed module Usage data and 13 months of aggregated product Usage data. If you wish to keep more than 3 months of detailed module Usage data, then run job HSISUSUM (Usage Summary) to summarize the detailed module Usage data older than three months. See job HSISUSUM for more details.

If you have not run the Usage Deletion job for some time, then select a period of a few months, in order to keep the run times down to a reasonable time.

Running Usage Deletion

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Usage Deletion. Member HSISUDEL contains JCL to run the job.

TPARAM parameters

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

DATABASE=
Name of database. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository schema.

UDELINVID=
Value must be specified. Inventory ID where you need to delete Usage data from. If UDELINVID=0, then Usage data from all Inventories are selected for deletion.

KEEPDETAIL=
Default is 2. Number of months prior to the current month for which detailed and summarized module Usage data are kept (and shown in Tivoli Common Reporting Discovery reports). KEEPDETAIL=0 means all

detailed and summarized module Usage data excluding those from the current month are deleted for the specified UDELINVID.

KEEPAGGR=
: Default is 12. Number of months prior to the current month for which aggregated product Usage data are kept (and shown in Tivoli Common Reporting Asset reports). KEEPAGGR=0 means all aggregated product Usage data, excluding those from the current month are deleted for the specified UDELINVID.

FIRSTDATE=
: Start of the first date range. This is in the form YYYYMM. Only complete months are chosen.

LASTDATE=
: End of the last date range. This is in the form YYYYMM.

  Note: The date range of deletion is inclusive of the month specified in the FIRSTDATE and LASTDATE parameters.

COMMIT=
: Default is 1000. Number of records stored before issuing of COMMIT.

Note: There are two options in managing deletion of detailed Usage data. If KEEPDETAIL is set to a value, then FIRSTDATE / LASTDATE will be ignored. If detailed Usage data are to be deleted within a certain date range, then comment out KEEPDETAIL and define dates for FIRSTDATE / LASTDATE. For further details, please see comments described in job HSISUDEL.

Delete Inventory

Delete Inventory is a facility where you can delete an individual Inventory from a Repository. A Repository is made up of many individual inventories that do, or do not, contain Usage data. Deleting an Inventory means all data associated with an Inventory is deleted. This includes Usage data if it has been imported.

It is recommended that you run this job on a periodic basis to minimize space utilization and improve SQL query performance.

Running Delete Inventory

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Delete Inventory utility. Member HSISDINV contains JCL to run the job.

TPARAM parameters

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
: Repository schema.

DELINVID=
: Value must be specified. Inventory ID to be deleted.

High-level qualifier listing for Usage Monitor

This job creates a high-level qualifier listing for Usage Monitor.

With the large amounts of Usage Data that are collected, it is best practice to monitor libraries that are identified in the high-level qualifier listing.

This utility scans the Repository to create a list of high-level qualifiers for products that are to be identified. Listed here are some examples that exclude all usage, but include some usage for the specified high-level qualifiers:

```
XDS(*)
IDS(DB2.*)
IDS(IMS.*)
IDS(CICS.*)
IDS(SYS1.*)
```

This process is currently automated in the Load to Repository job. The high-level qualifier listing is written to a data set, and this data set is concatenated to the Usage Monitor control file, HSIZIN.

Running high-level qualifier for Usage Monitor

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the High Level Qualifier for the Usage Monitor utility. Member HSISLLST contains JCL to run the job.

TPARAM parameters

DSN= DB2 location. Value assigned, as defined in HSISCUST.

REPSCHEMA=
Repository schema.

COMMIT=
Default is 1000. Number of records stored before issuing a COMMIT.

Repository Merge

Repository Merge is a facility where the Source Repository is merged or combined with another existing Repository, known as the Destination Repository. Running the Repository Merge facility repeatedly, with the same destination Repository, but with different source repositories, enables the combining of several separate repositories into one.

Destination Repository

The Destination Repository after a merge contains all the information it contained prior to the merge, plus all the information contained in the Source Repository.

Some particular cases are:

- If the Destination Repository does not already contain an Inventory with the same name as an Inventory in the Source Repository, then that entire Inventory is copied to the Destination Repository. This includes all libraries, all modules, all products and versions, and all Usage associated with that Inventory.
- If the Destination Repository does contain an Inventory with the same name as an Inventory in the Source Repository, then any additional libraries, modules, products, and versions, are copied to the relevant Inventory in the Destination Repository.
- Any Usage data for this Inventory in the Source Repository is copied across to the Destination Repository. If a Usage event for an identical library, module, product, version, user, job, and LPAR already exists in the Destination Repository, then the event count is summed. Otherwise, a new Usage record is

created. Statistics such as "last Usage date" are updated in order to reflect the later date of the source and the original Destination Repositories.
- Repository Merge merges all inventories in the Source Repository into the Destination Repository.
- Another function of this facility is its ability to rebuild the summary tables in a Repository. To do this, you must point the SRCSCHEMA and REPSCHEMA to the same Repository schema. Note that this facility is included as an extra job step in the Usage Import job, whereby summary tables are rebuilt after the Usage data have been imported.

Running Repository Merge

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the Repository Merge function. Member HSISMERG contains JCL to run the job.

TPARAM parameters

SSID= DB2 subsystem name. Value assigned, as defined in job HSISCUST.

DSN= DB2 location. Value assigned, as defined in job HSISCUST.

SRCSCHEMA=
Schemas for the source Repositories.

REPSCHEMA=
Schema for the target Repository.

DBNAME=
Name of Tivoli Asset Discovery for z/OS database. Value assigned, as defined in job HSISCUST.

COMMIT=
Default is 1000. Number of records stored before issuing a COMMIT.

TPARAM table update

In certain situations, the TPARAM table in the Repository might have been set to an inconsistent state, due to failures in jobs that update the Repository tables. To rectify this inconsistent state, a parameter in the TPARAM table has to be reset.

Running TPARAM table update

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the TPARAM update. Member HSISTPRM contains JCL to run the job.

SYSIN parameter

```
UPDATE &REPSCHMA.TPARAM SET FVALUE = '0' WHERE FKEY = 'PROCRUN';
```

SCRT import utility

The Tivoli Asset Discovery for z/OS SCRT import utility reads data created by the IBM Subcapacity Reporting Tool. This comes in the form of CSV formatted output files, and is used to populate several DB2 tables in the Repository. This information can then be queried by Tivoli Common Reporting to provide trending of SCRT data, as well as comparing this with the corresponding Usage Data.

Running SCRT import utility

The HSISCUST post-installation customization process creates a Job in the JCLLIB dataset to run the Tivoli Asset Discovery for z/OS SCRT Utility. Member HSISSCRT contains the JCL to run the job.

INPUT

DDNAME CSVIN, which contains the CSV output from the IBM SCRT tool. This may be from a data set with DSORG of PS or PO. Binary uploaded CSV files are supported.

DDNAME SIDMAP is used to map duplicate SMFIDs to a unique SID. The SCRT Import Utility is used to handle data where the same SMFID is used on multiple machines concurrently.

Map SMFID on specific machines to your desired SID. As described in this example, when processing data for CPU serial 11111, SMFID IP01, use SID QIP1, and so on.

```
//SIDMAP   DD *
11111-IP01=QIP1
11111-IP02=QIP2
11111-IP03=QIP3
/*
```

CPU serial
5 alphanumeric characters

SMFID
1 to 4 alphanumeric characters

Unique SID
1 to 4 alphanumeric characters. This must be the same as the SID value being used by the Usage Monitor for that z/OS system.

OUTPUT

Several DB2 tables are populated from the data contained in CSVIN, including NODE, NODE_CAPACITY, and PRODUCT_NODE_CAPACITY. Ensure that the CSVIN DD points to the .CSV output file created by the IBM SCRT tool. This may be a DSORG=PO or PS data set.

TPARAM parameters

SSID=
DB2 subsystem name. Value assigned, as defined in job HSISCUST

REPSCHEMA=
Repository schema name

GKB=
Default is GKB7. Schema name for the Global Knowledge Base

XML export utility

This utility extracts information in XML format for customers to import into the Tivoli Asset Management for IT product.

Running XML export utility

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the XML Export utility. Member HSISKBT contains JCL to run the job.

The output XML file generated from this utility needs to be ftp to a distributed environment and then loaded into Tivoli Asset Management for IT (TAMIT). Also the XML file needs to be translated from EBCDIC to ASCII. Here is an example on how to perform the ftp process.

1. C:\temp ftp *host name* (Using command line FTP connect to the host system)
2. When prompted enter your user name and password.
3. ftp > quote type a (set to ASCII)
4. ftp > get 'XML.EXPORT.FILE' C:\XML.FILE

   (ftp file from host to local workstation)
5. ftp > exit

   If you need to ftp non ASCII characters, then issue "ENCODING" statement as this extra command before the GET command :

   quote site ENCODING=MBCS MBDATACONN=(IBM-939,UTF-8)

   (this example is for a Japanese codepage)

TPARAM parameters

SSID=
DB2 subsystem name. Value assigned, as defined in job HSISCUST.

SCHEMA=
Repository schema.

a) Using the Repository schema value means that a catalog of products installed on your site is selected

b) You can also use the GKB schema value. This would mean that a catalog of all products defined in the GKB is selected.

HSISZCAT

The Usage Monitor produces at least one raw Usage data set each day. The ZCAT Utility (PGM=HSICZCAT) can be used to concatenate multiple Usage Monitor data sets into one single data set on the mainframe.

HSISZCAT scans for data sets with a predetermined naming convention using the following dsn construct:

```
umon-dsn-prefix.dyyyyddd.thhmmsst
```

The umon-dsn-prefix is determined by the UMDSN parm value. For example, PGM=HSICZCAT, PARM='UMDSN(umon-dsn-prefix)' . This is the same as the value defined on the Usage Monitor DSN(...) parameter. It has a maximum of 26 characters, which does not include the .dyyyyddd.thhmmsst suffix.

The data sets are logically concatenated to a single output data set and the input files are renamed to:

```
umon-dsn-prefix.dyyyyddd.shhmmsst
```

Note the letter 's' instead of 't' in the low level qualifier.

The output file may be pre-allocated. For example:

```
//ZCATOUT  DD DSN=your-dsname,RECFM=VB,LRECL=27994,UNIT=SYSALLDA,
//      SPACE=(CYL,(20,10),RLSE),DISP=(,CATLG)
```

or

the output data set is dynamically allocated with the following name:

```
umon-dsn-prefix.dyyyyddd.uhhmmsst
```

Note the use of the letter 'u' in the low level qualifier.

Running HSISZCAT

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run the concatenation of Usage Monitor data sets. Member HSISZCAT contains JCL to run the job.

Check the SYSPRINT DD for messages, output similar to this is displayed on execution:

```
DELETE option specified in the Parm string
input dataset identified: 'MACNIVE.MONITOR.FAE3.D2009105.T0000000'
input dataset identified: 'MACNIVE.MONITOR.FAE3.D2009106.T0000001'
input dataset identified: 'MACNIVE.MONITOR.FAE3.D2009107.T0000004'
Info: Catalog CATALOG.USER.SYSPLEXE had 3 dsnames.
Info: 3 'MACNIVE.MONITOR.FAE3.D%%%%%%%.T%%%%%%%' entries found
Info: 3 datasets ready for input processing

OPEN OUTPUT aok.  Dataset name= 'DD:ZCATOUT '
Open input dataset 'MACNIVE.MONITOR.FAE3.D2009105.T0000000' aok
Open input dataset 'MACNIVE.MONITOR.FAE3.D2009106.T0000001' aok
Open input dataset 'MACNIVE.MONITOR.FAE3.D2009107.T0000004' aok
ZCAT: shall now delete input datasets
DELETE 'MACNIVE.MONITOR.FAE3.D2009105.T0000000'
DELETE   'MACNIVE.MONITOR.FAE3.D2009106.T0000001'
DELETE 'MACNIVE.MONITOR.FAE3.D2009107.T0000004'
ZCAT: 3 datasets successfully logically concatenated to
      newly created 'DD:ZCATOUT ' dataset
```

Return Codes:

RC=00	ok
RC=01	No eligible input
RC=04	Some output okay with warnings
RC> 4	Serious error

Deleting HSISZCAT Input data sets

You can delete renamed input data sets by running HSISZCAT using the DELETE parameter. For example:

```
   //ZCAT EXEC PGM=HSICZCAT,PARM='UMDSN(umon.dsn.prefix),DELETE'
```

Chapter 16. DB2 performance considerations

There are various ways DB2 can be tuned to improve the performance of this product. They are described here in some detail.

In order to tune DB2 to improve the performance of this product, see the guidelines listed here. It is recommended that you implement this product in DB2 Version 9 in order to make use of some of the new features.

Choosing a DB2 subsystem for this product
: The DB2 resources required for this product do not need to be defined in a production DB2 subsystem. This is to minimize competition for mainframe resources in DB2 production environments. In addition, to avoid competing for mainframe resources, it is recommended that jobs for the Inquisitor Import, Match Engine, Load to Repository, and Usage Import, be run during off-peak periods. The utilities Usage Summary, Usage Deletion, and Delete Inventory, should also be run during off-peak periods.

HSISCUST
: To start, customize the parameters with optimal values in this job. Parameters that have a direct impact on the performance are **LOGGED** and variable buffer pool names. Note that parameter **LOGGED**, when set to ″N″, can only be used in Version 9. After implementation of the product, changes to parameters in DB2 objects can be accomplished using the ALTER TABLESPACE or ALTER INDEX DB2 commands.

Buffer pools
: By allocating the appropriate buffer pool to the respective table spaces and indexes, as defined in HSISCUST, you can manage your system resources accordingly. For DB2 performance, buffer pools should be investigated first. Check with your site specialist on the types and sizes of buffer pools that are defined for this product.

Space allocation and utilization
: In terms of space utilization, ″-1″ has been set for all SECQTY to enforce ″Sliding Secondary Extents″. This allows DB2 to manage secondary extents efficiently, and minimizes extension failures. You need to extrapolate the PRIQTY for the table spaces and indexes for the large tables according to your site’s requirements. Definitions for these DB2 objects are listed in the respective jobs in JCLLIB.

  Tables with the biggest impact on performance due to size alone are REPSchema.TMODULE, REPSchema.TUSEMTD, and REPSchema.TJOBDATA. Data for the REPSchema.TMODULE table is populated from the Inquisitor Import and Load to Repository processes, whilst REPSchema.TUSEMTD, and REPSchema.TJOBDATA tables are populated during the Usage Import.

  As an example, customers of very large sites can have more than 300 million rows in the REPschema.TUSEMTD table, and more that 5 million modules identified in the REPschema.TMODULE table.

IQ Import jobs
: For DB2 Version 9, it is recommended Inquisitor table spaces be defined with ″NOT LOGGED″. Data in Inquisitor tables are intermediate in nature and as such, logging is not necessary.

Declared Global Temporary tables
: Declared Global Temporary tables are used in the Match Engine job, and also during the Asset Aggregator process. The sizes of the TEMP table spaces must be sufficiently large to handle these two processes. In Version 8, DSNDB07 work file database does not have any 8K page size defined as part of DB2’s installation. If you are running in Version 8, you need to define at least one 8K table space in a TEMP database.

Work file database
: As some of the SQL queries, when run, can produce an excessive amount of output, it is recommended that the number and sizes of the table spaces in the work file database be increased.

Management of Usage data
: To minimize space utilization and improve SQL query performance, pruning of Usage data is necessary. Records from the two main tables, REPSchema.TUSEMTD and REPSchema.TJOBDATA, are to be deleted on a periodic basis.

Delete Inventory
: Sometimes there might be old inventories still in the system. To minimize space utilization, and improve performance for tables REPSchema.TMODULE, REPSchema.TJOBDATA, and REPSchema.TUSEMTD, it is recommended to delete obsolete inventories.

Reorganization and RUNSTATS
: It is important to run reorganization of the Repository table spaces periodically, especially after each Load to Repository, Usage data deletion, or Delete Inventory. After reorganization of the Repository table spaces, it is also a good idea to run RUNSTATS for these table spaces

Chapter 17. Deployment for large sites

It is recommended that each site has its own database and Tivoli Common Reporting server. A site in this context means a logical split. For example, a data center, geographical region, or outsourced customer. Having a database for each site avoids the need to regularly transfer high volume Usage data to a central site. It is also often easier to obtain DASD for several smaller databases than for one very large database.

The amount of data a DB2 subsystem can accommodate is only limited by availability of disk space. For example, if your site has 50 or 100 z/OS systems, you could load all your data into a single Repository. However, for data management and performance reasons, it is recommended you consider grouping your data into logical units. You could define multiple Repository schemas under a single database where each Repository schema might accommodate up to 30 systems. Alternatively, each Repository schema could be defined with its own database. For usability reasons, these Repository schemas are still collectively defined under the same DB2 subsystem. Further information on deployment scenarios can be found in Chapter 2, “Deployment scenarios,” on page 5.

In order to run Asset Reports that includes data from all sites, Asset table mirroring environment can be implemented. In summary:

- Asset tables from each site are mirrored to a central site.
- At the central site, a DB2 view is defined that is a union of the mirrored Asset tables.
- Tivoli Common Reporting is set up to report against the Asset mirror view, which includes product inventory and usage trends, but does not include details such as job names and libraries

Each site’s Repository has detailed discovery level data and aggregated asset level data. The Asset table mirroring only includes the asset level data, and therefore only requires small zip files to be transferred to the central site.

Running deployment for large sites

The HSISCUST post-installation customization job creates JCL jobs in the JCLLIB to run these jobs. Members HSISUNLD, HSISLOAD, HSISAM01 through to HSISAM04 contain JCL to run these jobs.

HSISUNLD
This job is used at each site to unload the Asset tables into a zip file. The zip file is then transferred using FTP to the central site for consolidation. Asset tables are a subset of the Repository tables, and these tables are unloaded for input to jobs HSISAM02 through to HSISAM04. It is also possible to unload all Repository tables and use the unloaded data for input into job HSISLOAD.

HSISAM01
This job defines the Asset table mirroring environment on the central site. It creates:

- Asset mirror tables for each mirrored site in schemas AM02, AM03 and AM04.

- Asset mirror view AM01, which is a union of the Asset tables of the local repository and the asset mirror schemas.

HSISAM02 through to HSISAM04
These jobs are used to load the zip files into the corresponding Asset table mirrors at the central site. The zipped unloaded data are obtained from HSISUNLD where they have been run separately for each site representing AM02, AM03, and AM04.

HSISLOAD
This job is provided as a sample for customers that may need to load data that were unloaded from HSISUNLD for tables found the Repository. You can either load data into all or some of the tables.

Note:

1. Jobs provided are to support data for 3 sites. If there are more than 3 mirrored sites, then job HSISAM04 can be cloned to HSISAM05 and so forth. Job HSISAM01 has to be changed to include additional tables for the DB2 views.
2. If there are less than 3 mirrored sites, then in job HSISAM01, remove DB2 resource definitions for the extra sites and change the DB2 views to remove the additional tables.

Tivoli Common Reporting for Asset mirror view

To set up Tivoli Common Reporting you create an extra copy of the reports which uses the new Asset mirror view as the source of data for the reports. You also require a TSO user ID and password that has read access to the Asset mirror view. Determine the JDBC URL needed to connect by running the HSISTCRR batch job to list the JDBC URLs available on DB2. The URL with currentSchema=AM01 will be the one needed in the data source.

1. Open the **Import Report Package** dialog and for **Local File** set HSITCR.zip as the report package. Under **Advanced Options**, set each option, **Report Set Base**, **Resource Directory Base**, **Design Name Base**, **Security Set**, **Namespace**, to the name to be displayed in the report set navigation tree, for example ABCD Enterprise. An extra copy of the reports will now appear in the navigation tree. From the new tree select **Asset Reports**, and right mouse click on one of the reports and choose **Data Sources.**
2. Edit the HSIzRep data source and set the user ID and password to the TSO user ID and password for the Asset mirror view. Set the JDBC URL to the URL with currentSchema=AM01 discovered from the HSISTCRR batch job and save.
3. Run the report to confirm the data source has been set up correctly. As only the Asset level reports have been combined in the mirror view, the other discovery report sets can be deleted. The discovery reports sets are **Discovery Administrator**, **Discovery Advanced**, and **Discovery Standard**. To delete these report sets, right mouse click on each of the reports and select **Delete**.

Appendix A. Messages

This section lists the various messages output by Tivoli Asset Discovery for z/OS

HSIA - Automation Server messages

Return codes

0	No errors encountered. All requests processed successfully.
16	Unrecoverable error. No requests processed. SYSIN or HSIPZIP or INQSOUT File cannot be used, or unsupported Operating System.

Message suffix codes

Suffix	Meaning	Raises Minimum Cond Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIA001E EXPECTED CLOSE PARENTHESIS WAS NOT FOUND IN INPUT RECORD

Explanation: Parsing did not detect the expected right parenthesis.

System action: Terminates the processing of the HSIAPARM member contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA002E EXPECTED OPERAND VALUE NOT FOUND BEFORE THE CLOSE PARENTHESIS

Explanation: No subparameter value was specified within the parenthesis.

System action: Terminates the processing of the HSIAPARM member contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA003E AN UNRECOGNISED OPERAND WAS ENCOUNTERED

Explanation: An operand other than TIME was specified.

System action: Terminates the processing of the HSIAPARM member contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA004E THE TIME OPERAND VALUE IS INVALID

Explanation: The TIME operand value did not conform to syntax requirements.

System action: Terminates the processing of the HSIAPARM member contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA005E NO ″FTP″ OR ″JOB″ STATEMENT BEFORE ″DSN″ STATEMENT

Explanation: There is no preceding action to relate the dsname mask to.

System action: Terminates the processing of the HSIAPARM member contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA006E AN UNRECOGNISED STATEMENT TYPE WAS ENCOUNTERED

Explanation: A statement type other than FTP, JOB or DSN was specified.

System action: Terminates the processing of the HSIAPARM member contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA007E TERMINATING - AUTOMATION SERVER IS ALREADY ACTIVE

Explanation: The Automation Server is already running. Only one concurrent copy can run in an operating system image.

System action: Program terminates with condition code 16. The established Automation Server continues.

User response: None.

Programmer response: None.

Module: HSIACUST

HSIA008E TERMINATING - COULD NOT INITIALISE WITH BAD PARAMETERS

Explanation: The HSIAPARM contents contained an error so the Automation Server could not initialize.

System action: The program terminates with condition code 16.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA009E REFRESH IGNORED - COULD NOT PROCESS BAD PARAMETERS

Explanation: The HSIAPARM contents contained an error so the Automation Server could not update its operational parameters.

System action: Terminates the processing of HSIAPARM contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA010E NO FUNCTIONS WERE REQUESTED

Explanation: No actions were specified. The Automation Server has no work to do.

System action: Terminates the processing of HSIAPARM contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA011E NO DATA SET NAME MASKS WERE SPECIFIED

Explanation: No work was requested. The Automation Server has no work to do.

System action: Terminates the processing of HSIAPARM contents.

User response: Correct the HSIAPARM member contents.

Programmer response: None.

Module: HSIACUST

HSIA012E NUMBER OF ACTIONS EXCEEDS MAXIMUM OF 32

Explanation: Too many actions were requested.

System action: Terminates the processing of HSIAPARM contents.

User response: Reduce the number of actions specified.

Programmer response: None.

Module: HSIACUST

HSIA013E NUMBER OF DATA SET NAME MASKS EXCEEDS THE MAXIMUM OF 250

Explanation: Too many dataset name masks were specified.

System action: Terminates the processing of HSIAPARM contents.

User response: Reduce the number of dataset name masks.

Programmer response: None.

Module: HSIACUST

HSIA014E MEMBER *1* WAS NOT FOUND IN THE HSIACNTL FILE

Explanation: Member HSIAPARM was found to be missing from the HSIACNTL file.

In the message text:

1 name of missing member.

System action: Terminates the processing of the member. If the member is HSIAPARM the Automation Server terminates. For template members the Automation Server continues processing.

User response: Create the required member in the HSIACNTL library.

Programmer response: None.

Module: HSIACUST

HSIA015E INPUT LOGICAL RECORD LENGTH WAS NOT 80

Explanation: A record was read from the HSIACNTL library which was not 80 bytes long.

System action: The program terminates and takes no actions.

User response: Ensure the HSIACNTL library only contains fixed length 80 byte records.

Programmer response: None.

Module: HSIACUST

HSIF - Conversion messages

Return codes

0	No errors encountered. All requests processed successfully.
4	I/O error in one or more program libraries.
8	Error - Incomplete data. Processing continues. OPEN or other system service error.
12	Syntax error.
16	Unrecoverable error.
20	Disastrous error. No requests processed. SYSPRINT file cannot be used.

Message suffix codes

Suffix	Meaning	Raises Minimum Condition Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIF000U PROCESSING TERMINATED - NO USABLE SYSPRINT FILE

Explanation: The OPEN of the SYSPRINT file failed.

Note: This message is issued by WTO with ROUTCDE=(2,11).

System action: Terminates with a condition code of 20.

Operator response: Ensure a usable SYSPRINT file is allocated.

System programmer response: None.

Module: HSIIM2D

HSIF001S PROCESSING TERMINATED - STORAGE OBTAIN ERROR ENCOUNTERED, RC=″*1*″

Explanation: Unable to acquire required amount of storage.

In the message text:

1 return code from STORAGE macro.

.

System action: Terminates with a condition code of 16.

Operator response: None.

System programmer response: Try increasing the region size specified in the region parameter on the JOB or EXEC statement in the JCL for the job. For addition information, examine the return code shown in the message and use for problem determination/resolution

Module: HSIIM2D

HSIF002S PROCESSING TERMINATED - DD FOR SYSPRINT MISSING

Explanation: DD statement missing for SYSPRINT. Note: This message is issued by WTO with ROUTCDE=(2,11).

System action: Terminates with a condition code of 20.

Operator response: Ensure DD statement in the JCL is correct.

System programmer response: None.

Module: HSIIM2D

HSIF004S PROCESSING TERMINATED - CANNOT OPEN ″*1*″ FILE

Explanation: The OPEN of the file failed.

In the message text:

1 name of file that failed the OPEN request.

.

System action: Terminates with a condition code of 16.

Operator response: Ensure a usable file is allocated.

System programmer response: None.

Module: HSIIM2D

HSIF005S PROCESSING TERMINATED - DD FOR FILENAME ″*1*″ MISSING

Explanation: DD statement missing for the file.

In the message text:

1 name of file with missing DD statement.

.

System action: Terminates with a condition code of 16.

Operator response: Ensure DD statement in the JCL is correct.

System programmer response: None.

Module: HSIIM2D

HSIF007S PROCESSING TERMINATED - INVALID SYSIN PARAMETER

Explanation: Parsing detected an invalid parameter

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF008S PROCESSING TERMINATED - INVALID SYSIN PLX= PARAMETER, MUST BE Y OR N

Explanation: The SYSIN PLX= parameter must be set to either Y or N

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF009S PROCESSING TERMINATED - INVALID SYSIN SYSPLEX= PARAMETER, THE SYSPLEX NAME MUST BE ENTERED

Explanation: The SYSIN SYSPLEX NAME must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF010S PROCESSING TERMINATED - INVALID SYSIN SSID= PARAMETER, THE SMF SYSTEM IDENTIFIER MUST BE ENTERED

Explanation: The SYSIN SSID= parameter must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF011S PROCESSING TERMINATED - INVALID SYSIN SSMF= PARAMETER, THE SMF SYSTEM IDENTIFIER MUST BE ENTERED

Explanation: The SYSIN SSMF= parameter must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF012S PROCESSING TERMINATED - INVALID SYSIN LPARNAME= PARAMETER, THE LPAR NAME MUST BE ENTERED

Explanation: The SYSIN LPAR NAME must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF013S PROCESSING TERMINATED - INVALID SYSIN SERIAL= PARAMETER, THE SERIAL NUMBER MUST BE ENTERED

Explanation: The SYSIN SERIAL NUMBER parameter must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF014S PROCESSING TERMINATED - INVALID SYSIN HWTYPE= PARAMETER, THE HARDWARE TYPE MUST BE ENTERED

Explanation: The SYSIN HARDWARE TYPE parameter must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF015S PROCESSING TERMINATED - INVALID SYSIN HWMODEL= PARAMETER, THE HARDWARE MODEL MUST BE ENTERED

Explanation: The SYSIN HARDWARE MODEL must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF016S PROCESSING TERMINATED - INVALID SYSIN UNM= PARAMETER, MUST BE Y OR N

Explanation: The SYSIN UNM= parameter must be set to either Y or N

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF017S PROCESSING TERMINATED - INVALID SYSIN JAC= PARAMETER, MUST BE Y OR N

Explanation: The SYSIN JAC= parameter must be set to either Y or N

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF018S PROCESSING TERMINATED - INVALID SYSIN PLANT= PARAMETER, THE PLANT MUST BE ENTERED

Explanation: The SYSIN PLANT must be entered

System action: Terminates with a condition code of 12.

Operator response: Correct the SYSIN file contents and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF019S PROCESSING TERMINATED - OUTPUT HEADER NOT CREATED, NO STATISTICS OR CAPACITY RECORD IN INPUT FILE

Explanation: The SYSIN PLANT must be entered

System action: Terminates with a condition code of 16.

Operator response: Ensure a usable input file exists and rerun the program.

System programmer response: None.

Module: HSIIM2D

HSIF020S PROCESSING TERMINATED - HSIMONZP *1* FAILURE, RC=″*2*″

Explanation: HSIMONZP file processing failure.

In the message text:

1 file processing in error.

2 return code from HSIMONZP.

.

System action: Terminates with a condition code of 16.

Operator response: Ensure the correct DD statements exist for the file and rerun the program. For additional information, examine the return code shown in the message and use for problem determination/resolution.

System programmer response: None.

Module: HSIIM2D

HSIF021S PROCESSING TERMINATED - NO DD STATEMENT FOUND FOR HSIINQOT OR HSIINQZP

Explanation: At least one of the DD statements must be included

System action: Terminates with a condition code of 16.

Operator response: Correct the JCL and rerun the program.

System programmer response: None.

Module: HSIIS2D

HSIF022S PROCESSING TERMINATED - *1* FAILURE, RC=″*2*″

Explanation: Processing failure.

In the message text:

1 processing in error.

2 return code.

.

System action: Terminates with a condition code of 16.

Operator response: None.

System programmer response: Contact IBM support.

Module: HSIIS2D

HSIF023S PROCESSING TERMINATED - HSIINQZP *1* FAILURE, RC=″*2*″

Explanation: HSIINQZP file processing failure.

In the message text:

1 file processing in error.

2 return code.

.

System action: Terminates with a condition code of 16.

Operator response: None.

System programmer response: Examine the return code shown in the message and use for problem determination/resolution

Module: HSIIS2D

HSIF024S **PROCESSING TERMINATED - HSIINQOT *1* FAILURE, RC=″*2*″**

Explanation: HSIINQOT file processing failure.

In the message text:

1 file processing in error.

2 return code.

.

System action: Terminates with a condition code of 16.

Operator response: None.

System programmer response: Examine the return code shown in the message and use for problem determination/resolution

Module: HSIIS2D

HSIF025S **PROCESSING TERMINATED - Table #LPRTBL# *1* FAILURE, RC=″*2*″**

Explanation: Table processing failure.

In the message text:

1 table operation in error.

2 return code.

.

System action: Terminates with a condition code of 16.

Operator response: None.

System programmer response: Examine the return code shown in the message and use for problem determination/resolution.

Module: HSIIS2D

HSIF999U **HSIMSG/HSIFMSG FAILURE - MSGID=*1*, RC=*2*, RS=*3***

Explanation: HSIMSG was called to produce a message text, but the call failed.

In the message text:

1 identifier of the failing message.

2 HSIMSG return code.

3 HSIMSG reason code.

.

System action: Terminates with a condition code of 20.

Operator response: Inform the system programmer.

System programmer response: Ensure Joblib/Steplib contains the library where the HSIFMSG message module resides, if you cannot resolve this issue then contact IBM support

Module: HSIIM2D

HSII - Utility messages

Return codes

0	No errors encountered. All requests processed successfully.
4	I/O error in one or more program libraries.
8	Error - Incomplete data. Processing continues. OPEN or other system service error.
12	Syntax error.
16	Unrecoverable error.
20	Disastrous error. No requests processed. SYSPRINT file cannot be used.

Message suffix codes

Suffix	Meaning	Raises Minimum Condition Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSII001I Read failed for SYSIN, RC=*1*

Explanation: The HSIICUST program could not read commands from the SYSIN DDname.

In the message text:

1 The return code from EXECIO

System action: The program terminates and takes no actions.

User response: Correct the JCL and provide a SYSIN DD with valid control statements.

Programmer response: None.

Module: HSIICUST

HSII002I Required Option *1* is missing from SYSIN

Explanation: The HSIICUST program did not find the required option in the SYSIN supplied by the user.

In the message text:

1 The name of the option that is missing.

System action: The program terminates and takes no actions.

User response: Correct the SYSIN and supply the option.

Programmer response: None.

Module: HSIICUST

HSII003I Optional Option *1* is missing from SYSIN, default *2* is provided.

Explanation: The HSIICUST program did not find an optional option in the SYSIN supplied by the user.

In the message text:

1 The name of the option that is missing.

2 The supplied default.

System action: The program continues and uses the default as documented in the message.

User response: Supply the option and re-run the customization program if the default is unacceptable.

Programmer response: None.

Module: HSIICUST

HSII004I Allocation of dataset *1* to *2* failed, RC=*3*

Explanation: The HSIICUST program could not allocate the dataset to the ddname for some reason.

In the message text:

1 The name of the dataset that failed allocation.

2 The DD name to be allocated.

3 The return code from the allocate command.

System action: The program terminates and takes no actions.

User response: Check the return code from the allocate command in the TSO commands manual. Correct the options and try running the program again. The probable option in error is HSIINST.

Programmer response: None.

Module: HSIICUST

HSII005I *1* failed for *2*, RC=*3*

Explanation: The HSIICUST program could not perform a required ISPF function because an error occurred during the ISPF function execution.

In the message text:

1 The name of ISPF function that failed.

2 The resource that caused the failure.

3 The return code from the ISPF function.

System action: The program terminates and takes no actions. The program may have written out JCL and parameter members. These members should be treated as suspect as they might contain errors in them due to the nature of this error.

User response: Check that the options specified do not exceed the field length requirements of the various options. If the problem persists, report the problem to IBM.

Programmer response: None.

Module: HSIICUST

HSII006I Model Dataset SHSISAMP has not been found.

Explanation: The HSIICUST program could not find the SHSISAMP dataset created during installation.

System action: The program terminates and takes no actions.

User response: Check that the HSI option is correctly specified and that the installation target libraries are available to the customization program.

Programmer response: None.

Module: HSIICUST

HSII010I ***1* statement verb not one of *1***

Explanation: During syntax parsing for a statement the verb found does not match any of the valid verbs for the program.

In the message text:

1 The word that is not a valid verb.

System action: The program terminates and takes no actions.

User response: Update the statements to the program to correct the verb in error and supply a correct verb.

Programmer response: None.

Module: HSIIREGN

HSII011I ***1* not valid for verb *1***

Explanation: During syntax parsing for a statement, a word was found that does not match the syntax of the statement for the verb that is being processed.

In the message text:

1 A word that is not valid for the statement syntax for a verb.

System action: The program terminates and takes no actions.

User response: Update the statements to the program to correct the statement in error.

Programmer response: None.

Module: HSIIREGN

HSII100I ***1* Missing from configuration.**

Explanation: The Region Assignment utility requires the parameter in the TPARAM dataset.

In the message text:

1 Parameter that is missing.

System action: The program terminates and takes no actions.

User response: Update the parameters in the TPARAM DD to add the missing parameter.

Programmer response: None.

Module: HSIIREGN

HSII101I **Attempt to define region *1* failed, region already exists.**

Explanation: The DEFINE REGION statement contains a region name that already exists.

In the message text:

1 Region to be defined.

System action: The program does not process this statement. Statements following the statement in error are not processed.

User response: Change the DEFINE REGION statement to refer to a region name that does not already exist.

Programmer response: None.

Module: HSIIREGN

HSII102I **Region *1* has not been defined.**

Explanation: The ASSIGN REGION statement contains a region name that does not exist.

In the message text:

1 Region to be assigned.

System action: The program does not process this statement. Statements following the statement in error are not processed.

User response: Change the ASSIGN REGION statement to refer to a region name that exists.

Programmer response: None.

Module: HSIIREGN

HSII103I **Inventory *1* has not been defined.**

Explanation: The ASSIGN INVENTORY statement contains an inventory that does not exist.

In the message text:

1 Inventory to be assigned.

System action: The program does not process this statement. Statements following the statement in error are not processed.

User response: Change the ASSIGN INVENTORY statement to refer to an inventory name that exists.

Programmer response: None.

Module: HSIIREGN

HSII104I **Region *1* is already assigned to *2***

Explanation: The ASSIGN REGION statement contains a region name that is to be assigned to another region, but the region being assigned is already assigned to a region.

In the message text:

1 Region to be assigned.

2 Parent region name.

System action: The program does not process this statement. Statements following the statement in error are not processed.

User response: Do not attempt to assign regions that are already assigned.

Programmer response: None.

Module: HSIIREGN

HSII105I Region *1* defined.

Explanation: The DEFINE REGION statement has executed correctly.

In the message text:

1 Region to be defined.

System action: The program continues with processing.

User response: None.

Programmer response: None.

Module: HSIIREGN

HSII106I Inventory *1* assigned to *2*

Explanation: The ASSIGN INVENTORY statement has executed correctly.

In the message text:

1 Inventory to be assigned.

2 Region name.

System action: The program continues with processing.

User response: None.

Programmer response: None.

Module: HSIIREGN

HSII107I *1* from *2* failed, RC=*3*

Explanation: An error has occurred executing the service for the resource specified. The service issued the return code mentioned.

In the message text:

1 Service that failed.

2 Resource that failed.

3 Return code from service.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Report this error to the System Programmer.

Programmer response: For ″EXECIO READ″ service, this probably means that the resource specified (ddname) is missing or empty. For all other error, report the problem to IBM Service.

Module: HSIIREGN

HSII108I SQL *1* failed, SQLCODE=*2*

Explanation: An error has occurred executing the SQL verb for the table specified.

In the message text:

1 SQL verb and table name.

2 SQLCODE from failing statement.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Report this error to the System Programmer.

Programmer response: Report the problem to IBM Service.

Module: HSIIREGN

HSII109I Inventory *1* is already assigned to *2*

Explanation: An error has occurred executing the SQL verb for the table specified.

In the message text:

1 Inventory name.

2 Region name.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Report this error to the System Programmer.

Programmer response: Report the problem to IBM Service.

Module: HSIIREGN

HSII110I Inventory *1* removed from *2*

Explanation: The inventory specified has been successfully removed from the region specified.

In the message text:

1 Inventory name.

2 Region name.

System action: The program continues processing.

User response: None.

Programmer response: None.

Module: HSIIREGN

HSII111I **Inventory *1* failed removal from *2*,SQLCODE=*3***

Explanation: The inventory specified has not been removed from the region specified.

In the message text:

1 Inventory name.

2 Region name.

3 SQL Return Code.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Report this error to the System Programmer.

Programmer response: Report the problem to IBM Service.

Module: HSIIREGN

HSII112I **Cannot delete region *1* - *2***

Explanation: The region specified must not be deleted from the database.

In the message text:

1 Region ID.

2 Region name.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Do not delete the region specified.

Programmer response: None.

Module: HSIIREGN

HSII113I **Cannot delete region *1*, region has children.**

Explanation: The region specified cannot be deleted because it has other regions assigned to it.

In the message text:

1 Region name.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Remove the region assignments from the region, before trying to delete the region.

Programmer response: None.

Module: HSIIREGN

HSII114I **Cannot delete region *1*, region has inventories assigned.**

Explanation: The region specified cannot be deleted because it has inventories assigned to it.

In the message text:

1 Region name.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Remove the inventory assignments from the region, before trying to delete the region.

Programmer response: None.

Module: HSIIREGN

HSII115I **Region *1* deleted.**

Explanation: The region specified has been deleted.

In the message text:

1 Region name.

System action: The program continues processing.

User response: None.

Programmer response: None.

Module: HSIIREGN

HSII116I **Region *1* not deleted, SQLCODE=*2***

Explanation: The region specified was not deleted due to an SQL error. SQLCODE details the error that occurred.

In the message text:

1 Region name.

2 SQL Return Code.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Report this error to the System Programmer.

Programmer response: Report the problem to IBM Service.

Module: HSIIREGN

HSII117I **Region *1* assigned to *2***

Explanation: The region specified was assigned to a parent region.

In the message text:

1 Region name.

2 Region name.

System action: The program continues processing.

User response: None.

Programmer response: None.

Module: HSIIREGN

HSII118I Region *1* contains DBCS characters.

Explanation: The region name specified must not contain DBCS characters.

In the message text:

1 Region name.

System action: The program stop processing statements. The current statement changes to DB2 tables are backed out.

User response: Do not put DBCS characters in the region name.

Programmer response: None.

Module: HSIIREGN

HSII200I *1* DDname CSVIN is missing.

Explanation: The SCRT Import utility requires the DDname CSVIN

In the message text:

1 DDname that is missing.

System action: The program terminates and takes no actions.

User response: Add the CSVIN DDname to the JCL and include the required input dataset containing data from the IBM SCRT tool.

Programmer response: None.

Module: HSIISCRT

HSII201I *1* CSVIN dataset is not PO or PS.

Explanation: SCRT Import Utility requires the CSVIN dsn to have a DSORG of PO or PS

In the message text:

1 CSVIN dataset not PO or PS.

System action: The program terminates and takes no actions.

User response: Ensure that the CSVIN dsn has a DSORG of PO or PS before submitting the Job.

Programmer response: None.

Module: HSIISCRT

HSII202I *1* contact IBM for a GKB refresh.

Explanation: The SCRT Import utility has detected a missing PID

In the message text:

1 PID is missing.

System action: The program continues.

User response: Contact IBM for a Global Knowledge Base (GKB) refresh. Include the missing PID number.

Programmer response: None.

Module: HSIISCRT

HSII300I Error writing to *1*.

Explanation: The XML Export utility has a problem writing the XML file

In the message text:

1 Error writing to DDNAME

System action: The program terminates and takes no further actions.

User response: Check the Return Code and any preceding messages.

Programmer response: None.

Module: HSIIKBT

HSII301I Number of lines written to SWKBTXML DD is*1*.

Explanation: No. of Lines written to SWKBTXML DD by the XML Export utility.

In the message text:

1 Lines written to SWKBTXML DD.

System action: The program continues and takes no actions.

User response: Informational message, no action required.

Programmer response: None.

Module: HSIIKBT

HSII302I SQL warning for *1*.

Explanation: SQL warning from a command

In the message text:

1 SQL warning.

System action: The program continues.

User response: Issue a warning message if the SQLCODE is not equal to 0 or 100.

Programmer response: None.

Module: HSIIKBT

HSII303I **SQL error for *1*.**

Explanation: The XML Export Utility has encountered an error.

In the message text:

1 SQL Error.

System action: The program terminates and takes no actions.

User response: Examine the SQL return code to determine the cause of the error. Contact IBM support with this information.

Programmer response: None.

Module: HSIIKBT

HSII304I **The XML Export Utility has encountered a DSNREXX error.**

Explanation: In the message text:

1 DSNREXX error.

System action: The program terminates and takes no actions.

User response: Examine the preceding error messages to determine the error. Contact IBM if you cannot resolve the problem

Programmer response: None.

Module: HSIIKBT

HSII305I **Invalid schema *1*.**

Explanation: The XML Export Utility has encountered a problem with an invalid Schema.

In the message text:

1 Invalid Schema

System action: The program terminates and takes no actions.

User response: Ensure that the correct Schema is being used.

Programmer response: None.

Module: HSIIKBT

HSIP - Inquisitor for z/OS messages and codes

Return codes

0	No errors encountered. All requests processed successfully.
4	I/O error in one or more program libraries.
8	Error - Incomplete data. Processing continues. OPEN or other system service error.
12	Syntax error.
16	Unrecoverable error. No requests processed. SYSIN or HSIPZIP or HSIPOUT File cannot be used, or unsupported Operating System.
20	Disastrous error. No requests processed. SYSPRINT file cannot be used.

Message suffix codes

Suffix	Meaning	Raises Minimum Condition Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIP000U **PROCESSING TERMINATED - NO USABLE SYSPRINT FILE**

Explanation: The OPEN of the SYSPRINT file failed. Note: This message is issue by WTO with

ROUTCDE=(2,11). All other messages are written to the SYSPRINT file.

System action: Terminates with a condition code of 20.

User response: Ensure a usable SYSPRINT file is allocated. The program overrides any unacceptable DCB values.

Programmer response: None.

Module: HSIPINQ

HSIP001U PROCESSING TERMINATED - CANNOT OPEN SYSIN FILE

Explanation: The OPEN of the SYSIN file failed.

System action: Terminates with a condition code of 16.

User response: Ensure a usable SYSIN file is allocated.

Programmer response: None.

Module: HSIPINQ

HSIP004S PROCESSING TERMINATED - UNKNOWN VERB *"1"*

Explanation: Parsing detected unrecognized data when looking for a verb.

In the message text:

1 name of the encountered verb.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP005S PROCESSING TERMINATED - UNKNOWN OPERAND *"1"*

Explanation: Parsing detected unrecognized data when looking for an operand.

In the message text:

1 name of the encountered operand.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP006S PROCESSING TERMINATED - UNEXPECTED LEFT PARENTHESIS ENCOUNTERED

Explanation: Parsing detected a left parenthesis at an unexpected location.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP007S PROCESSING TERMINATED - UNEXPECTED RIGHT PARENTHESIS ENCOUNTERED

Explanation: Parsing detected a right parenthesis at an unexpected location.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP008S PROCESSING TERMINATED - EXPECTED LEFT PARENTHESIS MISSING

Explanation: Parsing did not detect the expected left parenthesis.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP009S PROCESSING TERMINATED - EXPECTED RIGHT PARENTHESIS MISSING

Explanation: Parsing did not detect the expected right parenthesis.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP010U PROCESSING TERMINATED, OPERATING SYSTEM NOT SUPPORTED - CODE *"1"*

Explanation: The value of the byte at CVTDCB was not X'9B'.

In the message text:

1 hexadecimal value of first byte of CVTDCB.

System action: Terminates with a condition code of 16.

User response: This version of the Inquisitor cannot be run on this Operating System. If necessary, contact IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP011I MISSING RIGHT PARENTHESIS ASSUMED

Explanation: End-of-file was detected for SYSIN before an expected right parenthesis was detected.

System action: The request is accepted and processing continues.

User response: Correct the SYSIN file contents to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP012S PROCESSING TERMINATED - MISSING OPERAND SUBPARAMETER FOR *1*

Explanation: A required subparameter of an operand was not specified.

In the message text:

1 name of the operand being processed.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP013S PROCESSING TERMINATED - E-O-F INSTEAD OF EXPECTED CONTINUATION

Explanation: End-of-file was detected for SYSIN instead of an expected record required to continue the current statement being parsed.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP014I COMPLETED REQUEST NUMBER *1* - PROCESSING STATISTICS ARE:

Explanation: Processing of a request has been completed. A HSIP015I message follows containing the statistics for the request.

In the message text:

1 sequence number of the request.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIPINQ

HSIP015I VOLUMES=*1* DATASETS=*2* BAD-D/S=*3* PROGRAMS=*4*

Explanation: Processing of a request has been completed. Statistics related to the request are shown.

In the message text:

1 count of volumes scanned for this request.

2 count of data sets successfully processed.

3 count of data sets which could not be processed.

4 count of programs processed for this request.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIPINQ

HSIP016I ACCEPTED REQUEST NUMBER *1*

Explanation: Parsing of a request has been completed successfully. The request is stored for subsequent processing.

In the message text:

1 sequence number of the request.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP017E **DYNALLOC FAILURE: RC=*1* ERROR=*2* INFO=*3* VOLUME=*4***

Explanation: A data set could not be dynamically allocated. See message HSIP080I for the name of the dataset that incurred the problem.

In the message text:

1 return code of the DYNALLOC macro.

2 dynamic allocation return code (DARC).

3 dynamic allocation information code.

4 volume serial number of the data set.

System action: Processing of this data set is terminated.

User response: If necessary, rerun when the file is available for use. Note: The meanings of many DARC values are usually available in Appendix A of the ISPF Tutorial.

Programmer response: None.

Module: HSIPINQ

HSIP018E **VTOC DYNALLOC FAILURE: RC=*1* ERROR=*2* INFO=*3* VOLUME=*4***

Explanation: A VTOC could not be dynamically allocated. macro.

In the message text:

1 return code of the DYNALLOC

2 dynamic allocation return code (DARC).

3 dynamic allocation information code.

4 volume serial number of the data set.

System action: Processing of this volume is terminated.

User response: If necessary, rerun when the VTOC is available for use to process this volume. Note: The meanings of many DARC values are usually available in Appendix A of the ISPF Tutorial.

Programmer response: None.

Module: HSIPINQ

HSIP020I ***1* INQUISITOR OUTPUT RECORDS WRITTEN**

Explanation: Processing has concluded and all data files have been closed.

In the message text:

1 number of records written.

System action: Termination continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP021S **PROCESSING TERMINATED - INVALID OPERAND SUBPARAMETER FOR *1***

Explanation: The specified subparameter of an operand was not valid.

In the message text:

1 name of the operand being processed.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP022W **I/O ERR MEMBER *1* IN *2***

Explanation: An I/O error was encountered while reading the contents of a load module.

In the message text:

1 name of the program being processed.

2 name of the data set being processed.

System action: Processing of this member continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP023E **ABEND *1* IN OPEN FOR *2***

Explanation: An abnormal end occurred while opening a data set.

In the message text:

1 hexadecimal system abend and reason

2 name of the data set being processed.

System action: Processing of this data set is terminated.

User response: None required, but you may wish to exclude the data set from processing, or correct the cause of the abend.

Programmer response: None.

Module: HSIPINQ

HSIP024S PROCESSING TERMINATED - BAD UCBSCAN RETURN CODE OF HEX *1*

Explanation: An unexpected return code was received from UCBSCAN.

In the message text:

1 hexadecimal return code of the

System action: Processing of volume scanning for this request is terminated.

User response: Rerun the program when no dynamic reconfiguration changes are being implemented.

Programmer response: None.

Module: HSIPINQ

HSIP025U PROCESSING TERMINATED - CANNOT OPEN HSIPOUT FILE

Explanation: The OPEN of the HSIPOUT file failed.

System action: Terminates with a condition code of 16.

User response: Ensure that the allocated HSIPOUT file is usable, or omit the HSIPOUT file in favor of using the HSIPZIP file.

Programmer response: None.

Module: HSIPINQ

HSIP026E I/O ERROR ENCOUNTERED READING VTOC OF VOLUME *1* ON DEVICE *2*

Explanation: An I/O error was encountered while reading a VTOC.

In the message text:

1 volume serial number being processed.

2 device number of the volume.

System action: Processing of this track of the VTOC is terminated.

User response: None required, but you may wish to exclude the volume from processing, or correct the cause of the I/O error.

Programmer response: None.

Module: HSIPINQ

HSIP028U PROCESSING TERMINATED - CANNOT OPEN HSIPDMP FILE

Explanation: The OPEN of the HSIPDMP file failed after DUMPTEXT was specified.

System action: Terminates with a condition code of 16.

User response: Ensure a usable HSIPDMP file is allocated, or remove all DUMPTEXT operands from the contents of the SYSIN file. The DUMPTEXT operand should only be specified at the request of IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP029I TEXT-DUMPS=*2*

Explanation: Processing of a request with DUMPTEXT specified has completed. This message follows HSIP015I.

In the message text:

2 count of load module text blocks written.

System action: Processing continues.

User response: None required. The DUMPTEXT operand should only be specified at the request of IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP030I ″DUMPTEXT″ OPERAND IGNORED FOR ″SCANDIR″ VERB

Explanation: A DUMPTEXT operand was encountered for a SCANDIR request. That is, the possible dumping of load module text blocks was specified in a request which does not have access to text blocks.

System action: The DUMPTEXT operand is ignored and processing continues.

User response: Remove the DUMPTEXT operand to avoid this message. The DUMPTEXT operand should only be specified at the request of IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP031I BAD SELECTION CRITERIA FOR *2*

Explanation: Processing of a data set was specified but attributes did not match other selection criteria also specified in the request. This message is followed by HSIP038I which details the cause.

In the message text:

2 name of the data set being processed.

System action: Processing of this data set is terminated.

User response: If this data set is a program library which should be processed by the Inquisitor then modify or remove the conflicting selection criteria.

Programmer response: None.

Module: HSIPINQ

HSIP032I OBTAIN FAILED RC=*1* VOLUME *2*

Explanation: The system could not read the VTOC entry for the data set named in the HSIP033I message which follows this message. This message is only issued when a program parameter of ″DSNMSG″ or ″ALLMSG″ is specified.

In the message text:

1 hexadecimal return code of the OBTAIN macro.

2 volume serial number being processed.

System action: Processing of this data set is terminated.

User response: Ensure the relevant catalog entry is correct. Ensure the relevant volume is online and available to the system. Ensure there is no I/O error in the relevant volume's VTOC. Contact System Support for further assistance in diagnosis.

Programmer response: None.

Module: HSIPINQ

HSIP033I OBTAIN FAILED FOR DATASET *1*

Explanation: The system could not read the VTOC entry for the data set on the volume named in the previous HSIP032I message. This message is only issued when a program parameter of ″DSNMSG″ or ″ALLMSG″ is specified.

In the message text:

1 name of the data set being processed.

System action: Processing of this data set is terminated.

User response: Ensure the relevant catalog entry is correct. Ensure the relevant volume is online and available to the system. Ensure there is no I/O error in the relevant volumes VTOC. Contact System Support for further assistance in diagnosis.

Programmer response: None.

Module: HSIPINQ

HSIP034I REFER DATE WAS *1* FOR *2*

Explanation: A load library was opened. The reference date of the data set before the OPEN is reported in this message. This message is only issued when a program parameter of ″DSNMSG″ or ″ALLMSG″ is specified.

In the message text:

1 the Julian reference date from the VTOC entry.

2 name of the data set being processed.

System action: Processing of this data set continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP036E OPEN ERROR ENCOUNTERED READING VTOC OF VOLUME *1* ON DEVICE *2*

Explanation: The VTOC of the volume shown could not be opened.

In the message text:

1 volume serial number being processed.

2 device number of the volume.

System action: Processing of this track of the VTOC is terminated.

User response: None required, but you may wish to exclude the volume from processing, or correct the cause of the I/O error.

Programmer response: None.

Module: HSIPINQ

HSIP037E SECURITY ACCESS DENIED FOR *1*

Explanation: A RACROUTE macro determined the program had insufficient security access to read the data set.

In the message text:

1 name of the data set being processed.

System action: Processing of this data set is terminated.

User response: Contact Security Administration to obtain sufficient security access to read the data set or exclude the data set from processing.

Programmer response: None.

Module: HSIPINQ

HSIP038I BAD SELECTION CRITERIA WAS *1*

Explanation: Processing of a data set was specified but attributes did not match other selection criteria also specified in the request. This message follows HSIP031I which shows the data set name.

In the message text:

1 cause of the data set processing failure.

System action: Processing of this data set is terminated.

User response: If this data set is a program library which should be processed by the Inquisitor then modify or remove the conflicting selection criteria.

Programmer response: None.

Module: HSIPINQ

HSIP039S PROCESSING TERMINATED - ALL POSSIBLE DATA SETS ARE EXCLUDED

Explanation: An exclusion mask has been specified which excludes all possible data sets included by a selection mask. Both masks are shown after this message.

System action: Terminates with a condition code of 12.

User response: Modify or remove the conflicting selection criteria.

Programmer response: None.

Module: HSIPINQ

HSIP040S PROCESSING TERMINATED - ALL POSSIBLE DASD VOLUMES ARE EXCLUDED

Explanation: An exclusion mask has been specified which excludes all possible DASD volumes included by a selection mask. Both masks are shown after this message.

System action: Terminates with a condition code of 12.

User response: Modify or remove the conflicting selection criteria.

Programmer response: None.

Module: HSIPINQ

HSIP041S PROCESSING TERMINATED - ALL POSSIBLE PROGS ARE EXCLUDED

Explanation: An exclusion mask has been specified which excludes all possible programs included by a selection mask. Both masks are shown after this message.

System action: Terminates with a condition code of 12.

User response: Modify or remove the conflicting selection criteria.

Programmer response: None.

Module: HSIPINQ

HSIP042S PROCESSING TERMINATED, ALL POSSIBLE MODULES ARE EXCLUDED

Explanation: An exclusion mask has been specified which excludes all possible modules included by a selection mask.

System action: Terminates with a condition code of 12.

User response: Modify or remove the conflicting selection criteria. If no CSECT-level records are required then omit both MODULE and XMODULE operands.

Programmer response: None.

Module: HSIPINQ

HSIP043I ″MODULE″/″CSECT″ OPERAND IGNORED FOR ″SCANDIR″ VERB

Explanation: A MODULE operand was encountered for a SCANDIR request. That is, the output of program structure data was requested in a request which does not have access to this data.

System action: The MODULE operand is ignored and processing continues.

User response: Remove the MODULE operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP044I ″XMODULE″/″XCSECT″ OPERAND IGNORED FOR ″SCANDIR″ VERB

Explanation: An XMODULE operand was encountered for a SCABDIR request. That is, the output of program structure data was implied in a request which does not have access to this data.

System action: The XMODULE operand is ignored and processing continues.

User response: Remove the XMODULE operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP045I THE ″XDSNAME″ MASK IS NOT A SUBSET OF ANY ″DSNAME″ MASK

Explanation: The mask specified in the XDSNAME operand excludes possible values not included in the DSNAME operand. This message is issued to highlight possible inconsistencies in a request.

System action: Processing continues.

User response: Recode the XDSNAME operand as a further qualification of the DSNAME operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP046I THE ″XVOLUME″ MASK IS NOT A SUBSET OF ANY ″VOLUME″ MASK

Explanation: The mask specified in the XVOLUME operand excludes possible values not included in the VOLUME operand. This message is issued to highlight possible inconsistencies in a request.

System action: Processing continues.

User response: Recode the XVOLUME operand as a further qualification of the VOLUME operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP047I THE ″XPROGRAM″ MASK IS NOT A SUBSET OF ANY ″PROGRAM″ MASK

Explanation: The mask specified in the XPROGRAM operand excludes possible values not included in the PROGRAM operand. This message is issued to highlight possible inconsistencies in a request.

System action: Processing continues.

User response: Recode the XPROGRAM operand as a further qualification of the PROGRAM operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP048I THE ″XMODULE″ MASK IS NOT A SUBSET OF ANY ″MODULE″ MASK

Explanation: The mask specified in the XMODULE operand excludes possible values not included in the MODULE operand. This message is issued to highlight possible inconsistencies in a request.

System action: Processing continues.

User response: Recode the XMODULE operand as a further qualification of the MODULE operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP049I THE ″XSTOGROUP″ MASK IS NOT A SUBSET OF ANY ″STOGROUP″ MASK

Explanation: The mask specified in the XSTOGROUP operand excludes possible values not included in the STOGROUP operand. This message is issued to highlight possible inconsistencies in a request.

System action: Processing continues.

User response: Recode the XSTOGROUP operand as a further qualification of the STOGROUP operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP050I MODULES=*1*

Explanation: Processing of a request with MODULE specified has completed. This message follows HSIP015I.

In the message text:

1 count of CSECTs processed in this request.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP051I ****** PARSE ONLY REQUEST PROCESSED - NO ACTION TAKEN **********

Explanation: Processing of a SCANCMD request has completed.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP052U PROCESSING TERMINATED - MISSING HSIPOUT AND HSIPZIP FILES

Explanation: Neither an HSIPOUT nor an HSIPZIP file is allocated. At least one output file is required.

System action: Terminates with a condition code of 16.

User response: Specify an output file and rerun the job.

Programmer response: None.

Module: HSIPINQ

HSIP053U PROCESSING TERMINATED - COMPRESSION SUBROUTINE ERROR

Explanation: While processing the HSIPZIP file the compression subroutine encountered an error. The error message from the compression subroutine immediately follows this message.

System action: Terminates with a condition code of 16.

User response: Correct the error described in the message from the compression subroutine. If necessary, contact IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP054I ″FULLDIR″ OPERAND IGNORED FOR ″SCANDIR″ VERB

Explanation: A FULLDIR operand was encountered for a SCANDIR request. That is, the processing of load module member data was specified in a request which does not have access to this data.

System action: The FULLDIR operand is ignored and processing continues.

User response: Remove the FULLDIR operand to avoid this message.

Programmer response: None.

Module: HSIPINQ

HSIP056I *1 2* **COMMENCING SCAN OF VOLUME** *3* **ON UNIT** *4*

Explanation: A request without the CATALOG keyword began processing a DASD volume. This message provides feedback on the progress of long-running Inquisitor requests.

In the message text:

1 Date of message.

2 Time of message.

3 Serial number of volume being processed.

4 Device number of the volume.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP057E ABEND *1* **IN OPEN FOR VTOC OF VOLUME** *2* **ON UNIT** *3*

Explanation: A request without the CATALOG keyword attempted to open a DASD volume VTOC and the OPEN abended. The volume is not usable.

In the message text:

1 Hexadecimal system abend and reason codes.

2 Serial number of volume being processed.

3 Device number of volume being processed.

System action: Processing of this volume is terminated.

User response: Vary the volume offline, and/or reformat the volume. Institute any appropriate volume recovery procedures.

Programmer response: None.

Module: HSIPINQ

HSIP058S PROCESSING TERMINATED - DUPLICATE OPERAND ENCOUNTERED: *1*

Explanation: An input request was found to have the indicated operand specified more than once.

In the message text:

1 name of the duplicate operand

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program.

Programmer response: None.

Module: HSIPINQ

HSIP059E BINDER FAILURE FOR MEMBER *1* **RC=***2* **REASON=***3*

Explanation: The Binder could not successfully process a member of a PDSE. The Binder Fast Data Access API return and reason codes provide more detailed indicators of the cause. If the reason code is 1080003A then ensure that your system has fixes for IBM APARs OA08880 and OA09299 applied.

In the message text:

1 name of the member being processed.

2 hexadecimal Binder FDA API return code.

3 hexadecimal Binder FDA API reason code.

System action: Terminates data collection for this member, writes out data already collected and continues processing the next member.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP060S PROCESSING TERMINATED - SYMBOL SUBSTITUTION FAILURE - ASASYMBP RC=*1*

Explanation: The system symbol substitution routine could not successfully perform symbol substitution. Data before and after substitution is shown in the SYSPRINT file.

In the message text:

1 hexadecimal return code.

System action: Terminates with a condition code of 12.

User response: Correct or remove the symbols in control statement input.

Programmer response: None.

Module: HSIPINQ

HSIP061I *1* **NON-REEDITABLE IN** *2*

Explanation: A program object in a PDSE was encountered which has been marked non-reprocessable by the program binder. This message is only issued when a program parameter of ″PGMMSG″ or ″ALLMSG″ is specified.

In the message text:

1	Name of program which is not reprocessable.
2	Name of the data set being processed.

System action: Further data collection for this member is terminated.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP062S **PROCESSING TERMINATED, THE CATALOG REQUEST NEEDS EXACTLY ONE DSNAME MASK**

Explanation: A request with the CATALOG operand either omitted the DSNAME operand or specified more than one DSNAME mask.

System action: Terminates with a condition code of 12.

User response: Correct the SYSIN file contents and rerun the program. To process multiple data set name masks via the CATALOG specify a separate Inquisitor request for each mask. There is no programmed limit to the number of requests which can be processed in a single Inquisitor run.

Programmer response: None.

Module: HSIPINQ

HSIP063S **PROCESSING TERMINATED, ALL POSSIBLE STORAGE GROUPS ARE EXCLUDED**

Explanation: An exclusion mask has been specified which excludes all possible storage groups included by the selection mask. Both masks are shown after this message.

System action: Terminates with a condition code of 12.

User response: Modify or remove the conflicting selection criteria.

Programmer response: None.

Module: HSIPINQ

HSIP064E **ABEND** *1* **FOR** *2* **IN** *3*

Explanation: A subtask processing a program object from a PDSE has abended. The abend probably occurred in the Program Binder API.

In the message text:

1	Hexadecimal system abend code.
2	Name of the member being processed.
3	Name of the data set being processed.

System action: Data collected for this member so far is retained. Other Data Management abends may follow, especially in CLOSE processing, which are unrecoverable and may abend the main Inquisitor task.

User response: Exclude the programs causing the failure and rerun the Job. Ensure the fix for IBM APAR OW47547 is applied to your system.

Programmer response: None.

Module: HSIPINQ

HSIP065S **PROCESSING TERMINATED - MCDS FILE FAILED VERIFICAION**

Explanation: The MCDS data definition (DD) was found to be unusable by the Inquisitor. One or more of the following is true: o The MCDS file could not be opened. Message HSIP066E follows. o The MCDS file is not a VSAM key-sequenced data set (KSDS) o The KSDS relative key position (RKP) is not zero (0). o The KSDS key length is not forty-four (44).

System action: Terminates with a condition code of 12.

User response: Either ensure that the Inquisitor has read access to DFHSM's MCDS, or recode the Inquisitor request(s) so that the MCDS is not required. MCDS access is required if either or both of the REMIGRATE and NOML2 keywords are specified.

Programmer response: None.

Module: HSIPINQ

HSIP066E **MCDS OPEN ERROR - RC=***1* **REASON=***2*

Explanation: The OPEN of the MCDS data definition (DD) was not successful.

In the message text:

1	VSAM OPEN hexadecimal return code.
2	VSAM OPEN hexadecimal reason code.

System action: Issues message HSIP065S and terminates with a condition code of 12.

User response: Either ensure that the Inquisitor has read access to DFHSM's MCDS, or recode the Inquisitor request(s) so that the MCDS is not required.

MCDS access is required if either or both of the REMIGRATE and NOML2 keywords are specified.

Programmer response: None.

Module: HSIPINQ

HSIP067E MCDS READ RC=*1* REASON=*2* FOR *3*

Explanation: The MCDS record of a data set cataloged on volume MIGRAT could not be read. Either the record is missing or there was an I/O error.

In the message text:

1 VSAM GET hexadecimal return code.

2 VSAM GET hexadecimal reason code.

3 name of data set cataloged on volume MIGRAT.

System action: Processing of this data set is terminated.

User response: If the data set is not really migrated then correct the catalog entry. If the MCDS is corrupt then begin recovery procedures.

Programmer response: None.

Module: HSIPINQ

HSIP068E CATALOG RC=*1* RSN=*2*,*3* *4*

Explanation: The Catalog Search Interface returned an entry which is flagged as being in error by Catalog Management.

In the message text:

1 Catalog Management decimal return code.

2 Catalog Management decimal reason code.

3 Catalog Management module identifier.

4 name of catalog entry in error.

System action: Processing of this data set is terminated.

User response: Correct the catalog entry. Refer to the System Messages manual for message IDC3009I to find out the meaning of the Catalog Management error codes.

Programmer response: None.

Module: HSIPINQ

HSIP069U PROCESSING TERMINATED - PROGRAM IS NOT APF AUTHORIZED

Explanation: The Inquisitor has determined that it is not running in an APF authorized environment, and PARM=NOAPF was not specified.

System action: Terminates with a condition code of 20.

User response: Ensure that the HSIPINQ program is run in an APF authorized environment, or specify PARM=NOAPF in the JCL.

Programmer response: None.

Module: HSIPINQ

HSIP070E BAD BLKSIZE AFTER OPEN FOR *1*

Explanation: A BPAM DCB was opened for the named PDS, but despite the VTOC entry indicating a suitable block size, the block size in the DCB after the OPEN was not positive.

In the message text:

1 name of the data set being processed.

System action: Processing of member contents for this data set is terminated to avoid an S002-30 abend.

User response: The PDS is probably corrupt and should be deleted. Recreate it from a backup if appropriate.

Programmer response: None.

Module: HSIPINQ

HSIP071W IGNORING INVALID DSNAME IN *1*

Explanation: The Catalog Search Interface (CSI) returned a data set name with invalid characters. Although VTOC entries can contain keys that are not valid data set names, such entries cannot be cataloged. Therefore the entry returned from the CSI does not represent an actual data set.

In the message text:

1 name of the catalog being processed.

System action: The returned catalog entry is discarded.

User response: Ensure that the named catalog is not corrupt and contains no invalid entries.

Programmer response: None.

Module: HSIPINQ

HSIP072I BYPASS PROCESSING DATA SET *1*

Explanation: The name of the data set indicated that it does not contain programs which would normally be executed, and therefore the Inquisitor skipped processing it. This message is only issued when a program parameter of ″DSNMSG″ or ″ALLMSG″ is specified.

In the message text:

1 name of the data set being bypassed.

System action: The data set is not opened, and no data from it is collected.

User response: None required, but if the data set must be processed then specify its name in an inclusion mask without any generic masking characters, either by adding this mask to the existing request, or by adding an additional request to the same Inquisitor run.

Programmer response: None.

Module: HSIPINQ

HSIP073I NO DATA WAS EXTRACTED FROM *1*

Explanation: The data set contained no members eligible for selection. This message is only issued when a program parameter of ″DSNMSG″ or ″ALLMSG″ is specified.

In the message text:

1 name of the processed data set.

System action: The data set was opened, but no data from it is collected.

User response: None required.

Programmer response: None.

Module: HSIPINQ

HSIP074S PROCESSING TERMINATED, ABRIN OR ABRPRINT FILES NOT ALLOCATED

Explanation: A request had ABRMIG and/or ABRARC specified but at least one of the required ABRIN and ABRPRINT files was not defined in the JCL.

System action: Terminates with a condition code of 12.

User response: Ensure the required files are pre-allocated for the Inquisitor.

Programmer response: None.

Module: HSIPINQ

HSIP075W FDRABR ABEND *1* CHECKING *2*

Explanation: An abend occurred during ABR processing while checking a data set which may have been archived.

In the message text:

1 hexadecimal system abend code.

2 name of the data set being processed.

System action: Processing of this data set is terminated.

User response: Ensure the catalog entry for the data set is correct.

Programmer response: None.

Module: HSIPINQ

HSIP076E BAD LOAD *1*-*2*: *3* *4*

Explanation: The Inquisitor attempted to load a product tag data module from the named data set, but LOAD issued the displayed abend code.

In the message text:

1 abend code returned by LOAD.

2 abend reason code returned by LOAD.

3 name of the member containing the tag data.

4 name of the data set containing the tag data module.

System action: Processing continues with the next member in the data set.

User response: Verify that the named data set contains no unusable modules. If necessary, delete any modules that are of no further use.

Programmer response: None.

Module: HSIPINQ

HSIP077E ISITMGD RC=*1* RSN=*2* FOR *3*

Explanation: The Inquisitor executed an ISITMGD macro for the named data set, but ISITMGD issued a non-zero return code.

In the message text:

1 decimal return code issued by ISITMGD.

2 hexadecimal reason code issued by ISITMGD.

3 name of the data set being processed.

System action: Processing continues with the next data set.

User response: Consult the applicable DFSMS Macro Instructions for Data Sets manual to determine the meaning of the ISITMGD return and reason codes. Ensure that the named data set is a valid and accessible partitioned data set. If necessary, gather appropriate diagnostic materials and contact IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP078E DESERV RC=*1* RSN=*2* FOR *3*

Explanation: The Inquisitor executed a DESERV FUNC=GET_ALL macro for the named data set, but DESERV issued a non-zero return code.

In the message text:

1 decimal return code issued by DESERV.

2 hexadecimal reason code issued by DESERV.

3 name of the data set being processed.

System action: Processing continues with the next data set.

User response: Consult the applicable DFSMS Macro Instructions for Data Sets manual to determine the meaning of the DESERV return and reason codes. Ensure that the named data set is a valid and accessible partitioned data set. If necessary, gather appropriate diagnostic materials and contact IBM support.

Programmer response: None.

Module: HSIPINQ

HSIP080I DYNALLOC FAILURE: DSN=*1*

Explanation: A data set could not be dynamically allocated.

In the message text:

1 name of the data set being processed.

System action: Depends upon other messages associated with this message.

User response: None.

Programmer response: None.

Module: HSIPINQ

HSIP097E CATALOG SEARCH INTERFACE ERROR RC=*1*

Explanation: A request with the CATALOG keyword was specified, and the Catalog Search Interface encountered an error.

In the message text:

1 return code from the Catalog Search Interface.

System action: Processing catalog entries for the request is terminated.

User response: Correct any related catalog errors.

Programmer response: None.

Module: HSIPINQ

HSIP098E CATALOG SEARCH INTERFACE ERROR RC=*1* CATALOG RC=*2* REAS=*3*

Explanation: A request with the CATALOG keyword was specified, and the Catalog Search Interface encountered an error.

In the message text:

1 return code from the Catalog Search Interface.

2 return code from Catalog Management.

3 reason code from Catalog Management.

System action: Processing catalog entries for the request is terminated.

User response: Correct any related catalog errors.

Programmer response: None.

Module: HSIPINQ

HSIP099E CATALOG SEARCH INTERFACE ERROR RC=*1* CATALOG RC=*2* REAS=*3* MODULE=*4*

Explanation: A request with the CATALOG keyword was specified, and the Catalog Search Interface encountered an error.

In the message text:

1 return code from the Catalog Search Interface.

2 return code from Catalog Management.

3 reason code from Catalog Management.

4 module identifier.

System action: Processing catalog entries for the request is terminated.

User response: Correct any related catalog errors.

Programmer response: None.

Module: HSIPINQ

HSIP999U MODULE HSIPMSG FAILED - MSGID=*1*, RC=*2*, REASON=*3*

Explanation: HSIMSG was called to produce a message text, but the call failed.

In the message text:

1 identifier of the failing message.

2 HSIMSG return code.

3 HSIMSG reason code.

System action: Terminates with a condition code of 20.

User response: Inform the system programmer.

Programmer response: Contact IBM Support.

Module: HSIPINQ

HSIT - Product Tagging messages

Return codes

0	No errors encountered. All requests processed successfully.

4	Warning issued. Processing continues. I/O error in one or more program libraries.
8	Error - Incomplete data. Processing continues. OPEN or system service error.
12	Severe error. Processing terminates. Utility failure or syntax error.
16	Unrecoverable error. No requests processed. SYSIN file cannot be used.
20	Disastrous error. No requests processed. SYSPRINT file cannot be used.

Message suffix codes

Suffix	Meaning	Raises Minimum Condition Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIT001U HSITAGP could not open the input file *1*

Explanation: A required file could not be opened successfully.

In the message text:

1 name of file.

System action: Processing terminates with condition code 16.

Operator response: None.

User response: Correct the file definition and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT002S Unrecognized statement type: *1*

Explanation: Input text was encountered which does not match any known statement type.

In the message text:

1 encountered input data.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT003S Duplicate value supplied for *1*

Explanation: More than one occurrence of the named statement type was encountered but only one value can be accepted.

In the message text:

1 name of the statement verb.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Remove the redundant statement and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT004S Value missing in *1* **statement**

Explanation: An input statement of the type indicated was encountered but no non-blanks followed the statement type name.

In the message text:

1 name of the statement verb.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Supply an appropriate value after the statement type name.

Programmer response: None.

Module: HSITAGP

HSIT005S Value specified for LICENSED was neither ″YES″ nor ″NO″

Explanation: A LICENSED statement was processed which had a value specified other than one of the valid values.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the value and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT006S The *1* parameter had no subparameter value specified

Explanation: A statement parameter or operand was specified but the required subparameter or value of the parameter was not specified. One cause for this condition is the omission of a parenthesis.

In the message text:

1 name of the parameter or operand being processed when the error is detected.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT007I A closing parenthesis assumed for *1*

Explanation: End-of-file was raised when processing input statements before an expected close parenthesis was encountered.

In the message text:

1 name of the parameter or operand being processed when the error is detected.

System action: Processing continues as if the expected close parenthesis had been specified.

Operator response: None.

User response: Check that the resulting processing is as expected. Correct the input file for future use, and rerun the job if the desired processing was not performed.

Programmer response: None.

Module: HSITAGP

HSIT008S Unexpected open parenthesis encountered after *1*

Explanation: An open parenthesis was encountered when one was not expected. If this occurred while a parameter or operand was being processed then it is named in the message.

In the message text:

1 name of the parameter or operand being processed when the error is detected.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input file and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT009S Unexpected close parenthesis encountered after *1*

Explanation: A close parenthesis was encountered when one was not expected. If this occurred while a parameter or operand was being processed, then it is named in the message.

In the message text:

1 name of the parameter or operand being processed when the error is detected.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input file and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT010S *1* is an unknown TAGLIBS parameter

Explanation: Input data was encountered which is not a recognized parameter or operand of the TAGLIBS statement.

In the message text:

1 the encountered input data.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input file and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT011S Member name *1* has embedded blank(s)

Explanation: The value specified on the TAGMEM statement was not a valid partitioned data set member blank within the 8-character member name.

In the message text:

1 the input value specified on the TAGMEM statement.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input file and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT012S Missing open parenthesis after *1*

Explanation: Whilst parsing the TAGLIBS statement looking for a subparameter or value in parentheses specified for the parameter or operand named in the message, text was encountered which was not enclosed in parentheses.

In the message text:

1 name of the parameter or operand being processed when the error is detected.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT013S Value *1* too long for parameter *2*

Explanation: The length of a subparameter or value was found to exceed the maximum length allowed. The maximum length allow depends on the specific parameter or operand being processed. For example, a data set name mask exceeding 44 characters in length causes this condition, as will a volume mask exceeding 6 characters in length.

In the message text:

1 encountered input data.

2 name of the parameter or operand being processed when the error is detected.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Correct the input and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT014S End of input reached, expected continuation is missing

Explanation: End-of-file was raised on the input (SYSIN) file, but the TAGLIBS statement currently being processed was expected to continue on the next record.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Either supply the missing input data or remove the continuation character from the last input record, and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT015S Required data set name specification is missing

Explanation: The processing of a TAGLIBS statement completed without encountering a data set name selection mask specification.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Ensure that all TAGLIBS statements specify at least one value in a DATASET or DSNAME parameter or operand.

Programmer response: None.

Module: HSITAGP

HSIT016S Symbol substitution failure - ASASYMBM RC=*1*

Explanation: Symbol substitution was attempted for a TAGLIBS statement record which had at least one ampersand in it, and the system symbol substitution routine ASASYMBM terminated with a non-zero return code.

In the message text:

1 the completion code returned by ASASYMBM.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Check all uses of symbols in the input (SYSIN) file. If necessary, gather appropriate diagnostic materials and contact IBM support.

Programmer response: None.

Module: HSITAGP

HSIT017S No value for *1* was specified

Explanation: A value for a statement of the type named in the message is required, but was not found in the input file.

In the message text:

1 the type of input statement required to specify the missing value.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Supply a statement of the named type which specifies a value.

Programmer response: None.

Module: HSITAGP

HSIT018W Catalog search terminated by RC=*1* while searching for *2*

Explanation: The Catalog Search Interface (CSI) was called to search for catalog entries matching the displayed data set name mask, but the CSI call ended with a non-zero return code. This may indicate a corrupt catalog or trouble accessing a potentially relevant catalog. It may mean that one or more data sets which you intended to process were not processed.

In the message text:

1 the return code issued by the Catalog Search Interface.

2 the data set name mask passed to the Catalog Search Interface.

System action: Processing continues with any data set catalog entries that can be accessed.

Operator response: None.

User response: Check the data set name mask for suitability to your system environment. Use IDCAMS to examine any relevant catalog entries. Ensure that all catalogs which would be referenced by such a search are on accessible volumes. Use the same data set name mask with other software which also calls the CSI such as the Inquisitor or ISPF option 3.4 to see if further diagnostic information can be acquired. The error is usually associated with the local catalog configuration and catalog contents.

Programmer response: None.

Module: HSITAGP

HSIT019E DESERV failed - RC=*1* Reason=*2* for data set *3*

Explanation: A DESERV FUNC=GET_ALL macro was issued to acquire the member list for a data set but DESERV issued a non-zero return code.

In the message text:

1 the decimal return code issued by DESERV.

2 the hexadecimal reason code issued by DESERV.

3 the name of the data set being processed by DESERV.

System action: The named data set is not processed, and processing continues with the next relevant data set, if any.

Operator response: None.

User response: Consult the applicable DFSMS Macro Instructions for Data Sets manual to determine the meaning of the DESERV return and reason codes. Ensure that the named data set is a valid and accessible program library. If necessary, gather appropriate diagnostic materials and contact IBM support.

Programmer response: None.

Module: HSITAGP

HSIT020S Dynamic allocation failure - BPXWDYN RC=*1*

Explanation: BPXWDYN was called to dynamically allocate a required work file but BPXWDYN issued a non-zero return code. As a result, processing cannot proceed.

In the message text:

1 the hexadecimal return code issued by BPXWDYN.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Consult the applicable Using REXX and z/OS UNIX System Services manual to determine the meaning of the return code. Examine the job log and messages to see any associated dynamic allocation error message.

Programmer response: None.

Module: HSITAGP

HSIT021E DYNALLOC failed - RC=*1* Error=*2* Infor=*3* for data set *4*

Explanation: A DYNALLOC macro was issued to dynamically allocate a program library for processing but DYNALLOC issued a non-zero return code.

In the message text:

1 the decimal return code issued by DYNALLOC.

2 the contents of S99ERROR in hexadecimal.

3 the contents of S99INFO in hexadecimal.

4 name of the data set which could not be allocated.

System action: Processing continues with the next data set, if any.

Operator response: None.

User response: Consult the applicable MVS Authorized Assembler Services Guide to determine the meaning of the dynamic allocation return, error, and information codes. Check that the named data set is accessible and available for allocation.

Programmer response: None.

Module: HSITAGP

HSIT022S Return code *1* was returned by program *2*

Explanation: Either the High Level Assembler (program ASMA90) or the Program Binder (program IEWL) was dynamically invoked to assist with creating the output data, but the named program issued a non-zero return code.

In the message text:

1 the decimal return code issued by the named program.

2 the name of the program that was invoked.

System action: Processing terminates with condition code 12.

Operator response: None.

User response: Examine all associated job output to determine if the problem is caused by a correctable environmental error. If so, make the correction and rerun the job. If not, gather all relevant diagnostic materials and contact IBM support.

Programmer response: None.

Module: HSITAGP

HSIT023I Processing terminated due to encountered error condition

Explanation: Because of a previously reported error, the Product Tagging Utility is terminating unilaterally, quite possibly without processing all of the specified program library data sets, and without generating all of the requested program product tagging data.

System action: Processing terminates.

Operator response: None.

User response: Investigate any previously reported error conditions as appropriate.

Programmer response: None.

Module: HSITAGP

HSIT024E ISITMGD failed - RC=*3* Reason=*4* for file *5* and data set *6*

Explanation: An ISITMGD macro was issued against a program library but ISITMGD issued a non-zero return code.

In the message text:

3 the decimal return code issued by ISITMGD.

4 the decimal reason code issued by ISITMGD.

5 the name of the file being processed by ISITMGD.

6 the name of the data set being processed by ISITMGD.

System action: The named data set is not processed, and processing continues with the next relevant data set, if any.

Operator response: None.

User response: Consult the applicable DFSMS Macro Instructions for Data Sets manual to determine the meaning of the ISITMGD return and reason codes. Ensure that the named data is a valid and accessible partitioned data set. If necessary, gather the appropriate diagnostic materials and contact IBM support.

Programmer response: None.

Module: HSITAGP

HSIT025I *1* programs found to tag from data set *2*

Explanation: Input processing of the named data set has completed, resulting in data from the reported number of programs being accumulated for subsequent output.

In the message text:

1 the number of programs processed.

2 the data set name containing the processed programs.

System action: Processing continues.

Operator response: None.

User response: None required.

Programmer response: None.

Module: HSITAGP

HSIT026I Processing complete - CC=*1* - *2* programs tagged in total

Explanation: The Product Tagging Utility program HSITAGP has completed processing. This message reports the condition code issued by HSITAGP and the number of programs from which data has been collected during this run.

In the message text:

1 the condition code issued by the HSITAGP upon termination.

2 the number of programs processed in this run of HSITAGP.

System action: Processing completes with the displayed condition code.

Operator response: None.

User response: None required.

Programmer response: None.

Module: HSITAGP

HSIT027E **DYNALLOC failed - RC=*1* Error=*2* Info=*3* for volume *4***

Explanation: A DYNALLOC macro was issued to dynamically allocate a volume for processing but DYNALLOC issued a non-zero return code.

In the message text:

1 the decimal return code issued by DYNALLOC.

2 the contents of S99ERROR in hexadecimal.

3 the contents of S99INFO in hexadecimal.

4 the volume serial number of the volume which could not be allocated.

System action: Processing continues with the next volume, if any.

Operator response: None.

User response: Consult the relevant MVS Authorized Assembler Services Guide to determine the meaning of the dynamic allocation return, error, and information codes. Check that the named volume is accessible and available for allocation.

Programmer response: None.

Module: HSITAGP

HSIT028W **Unable to acquire any product maintenance level date**

Explanation: After having processed all of the relevant programs HSITAGP was unable to acquire any date stamp for use as a maintenance level indicator.

System action: Blanks are placed in the maintenance level field and processing continues.

Operator response: None.

User response: None required.

Programmer response: None.

Module: HSITAGP

HSIT029S ***1* statement value length exceeds the allowed maximum of *2* bytes**

Explanation: The value specified for the named statement type was found to be longer than the maximum allowed. The maximum byte count allowed for a value of this statement type is shown in the message.

In the message text:

1 the type of input statement being processed.

2 number of bytes.

System action: Processing terminates.

Operator response: None.

User response: Correct the input and rerun the job.

Programmer response: None.

Module: HSITAGP

HSIT030S **Invalid text character X″*1*″ found in *2* statement**

Explanation: The displayed data byte was encountered when processing the value specified for the statement type indicated. The value specified on the statement is expected to be a string. Valid byte values for text data are in the range from X′40′ to X′FE′ inclusive. The control code encountered is either not valid input or not valid input in this location. The only control codes that may be used in the input value are SO (X′0E′) and SI(X′0F′) when they are used to encapsulate DBCS data.

In the message text:

1 the hexadecimal value of the invalid text code-point.

2 the type of input statement being processed.

System action: Processing terminates.

Operator response: None.

User response: Remove the undisplayable characters from the input value. If using DBCS, ensure that SO precedes DBCS text and SI terminates DBCS text, and that the DBCS text is an even number of valid text bytes.

Programmer response: None.

Module: HSITAGP

HSIX - Inquisitor for z/OS UNIX messages and codes

Return codes

0	No errors encountered. All requests processed successfully.
4	I/O error in one or more program libraries.
8	Error - Incomplete data. Processing continues. OPEN or other system service error.
12	Syntax error.
16	Unrecoverable error. No requests processed. SYSIN or HSIPZIP or HSIPOUT File cannot be used, or unsupported Operating System.
20	Disastrous error. No requests processed. SYSPRINT file cannot be used.

Message suffix codes

Suffix	Meaning	Raises Minimum Condition Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIX001I SPECIFIED DIRECTORY NAME IGNORED DUE TO LEADING BLANK

Explanation: A record from file HSIXROOT was read and was found to start with a blank.

System action: The program ignores the record.

User response: Correct the input. Directory names in HSIXROOT should start in column 1.

Programmer response: None.

Module: HSIXINQ

HSIX002I THE SPECIFIED DIRECTORY NAME DOES NOT START WITH A SLASH

Explanation: A record from file HSIXROOT was read and was found to start with a non-blank that is not a slash. It is flagged in case processing errors result from the non-standard directory name.

System action: Processing continues.

User response: Correct the input if it is incorrect.

Programmer response: None.

Module: HSIXINQ

HSIX003I PROGRAM PARAMETER "*1*" DISCARDED

Explanation: The program parameter contained some unrecognized data.

In the message text:

1 Parameter in error

System action: The displayed part of the program parameter is ignored.

User response: Correct the program parameter.

Programmer response: None.

Module: HSIXINQ

HSIX004I FUNCTION *1* FAILED, HEX RC=*2*, HEX REASON=*3*

Explanation: The named z/OS UNIX system service issued a negative return code.

In the message text:

1 Function name.

2 Return Code.

3 Reason Code.

System action: Processing continues.

User response: Determine the meaning of the return and reason code values, and correct the problem if appropriate.

Programmer response: None.

Module: HSIXCUST

HSIX005E *1*

Explanation: The named path was not successfully processed by the z/OS UNIX system service named in the preceding HSIX004I message. Data will not be collected from directories and files which could not be opened.

In the message text:

1 Path in error

System action: Processing continues.

User response: Ensure that all necessary parts of the z/OS UNIX file system are accessible.

Programmer response: None.

Module: HSIXCUST

HSIZ - Usage Monitor messages

Return codes

0	Normal termination.
16	Initialization failed.

Message suffix codes

Suffix	Meaning	Raises Minimum Cond Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIZ001I **USAGE MONITOR INITIALIZING**

Explanation: The Usage Monitor has been started.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ002I *1* **DETECTED UNSUPPORTED OPERATING SYSTEM**

Explanation: The Usage Monitor may not run on an unsupported operating system.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ003I *1* **USAGE MONITOR NOT APF AUTHORIZED**

Explanation: The Usage Monitor needs to be executed in an APF authorized environment.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: See System Programmer to correct the error.

Programmer response: APF authorize the load libraries that the Usage monitor runs from.

Module: HSIZMON

HSIZ005I *1* **USAGE MONITOR ALREADY ACTIVE**

Explanation: The Usage Monitor is already running. Only one concurrent copy can run in an operating system image.

In the message text:

1 current system identifier.

System action: Processing terminates. The established Usage Monitor task continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ006I *1* **USAGE MONITOR QEDIT BUFFER SET FAILED**

Explanation: A QEDIT issued to set up MODIFY command processing has failed.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: Notify the system programmer.

Programmer response: If necessary, contact IBM support.

Module: HSIZMON

HSIZ007I *1* **USAGE MONITOR MODULE** *2* **FAILED - RETURN CODE=***3*

Explanation: A Usage Monitor subroutine has failed.

In the message text:

1 current system identifier.

2 failing module name.

3 decimal return code.

System action: Processing terminates.

User response: Notify the system programmer.

Programmer response: If the return code is 312, then you must increase your MAXCAD parameter. This requires an IPL. For any other return codes, retain any diagnostic materials and contact IBM support.

Module: HSIZMON

HSIZ008I *1* **USAGE MONITOR ASID** *2* **SET IN AVT** *3*

Explanation: An Anchor Vector Table (AVT) has been acquired or reacquired, and has been updated for the current server address space.

In the message text:

1 current system identifier.

2 ASID number.

3 AVT Address.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ009I **DATA WRITTEN TO DSN=***1*

Explanation: Usage Monitor data has been written to the named data set.

In the message text:

1 data set name of the created output

System action: Processing continues.

User response: Transfer the named data set to the system where the database resides so it can be processed.

Programmer response: None.

Module: HSIZMON

HSIZ010E *1* **USAGE MONITOR - WRITER TASK ENDED - COND CODE** *2*

Explanation: A writer task has ended with a non-zero completion code.

In the message text:

1 current system identifier.

2 Return code of writer task.

System action: Processing continues.

User response: Notify the system programmer.

Programmer response: If necessary, contact IBM support.

Module: HSIZMON

HSIZ011E *1* **USAGE MONITOR - WRITER TASK ABENDED - S***2*

Explanation: A writer task has ended abnormally.

In the message text:

1 current system identifier.

2 Abend code from writer task.

System action: Processing continues.

User response: Notify the system programmer.

Programmer response: Local reasons for system abends should be investigated. If necessary, retain all diagnostic materials and contact IBM support.

Module: HSIZMON

HSIZ012I **DATA LOSS UNUSABLE DSN=*1***

Explanation: It is likely that Usage Monitor data has been lost because of unexpected behavior by a writer task. Any compressed output data that has been written will probably be unusable.

In the message text:

1 data set name of the created output file.

System action: Processing continues.

User response: Examine any preceding messages to determine the likely cause of the writer task error. If the output data set is complete it can be used, otherwise if the data is compressed it is unusable. If the data set is empty then this fact can be noted and the data set can be deleted. Unless retaining an unusable data set for diagnosis reasons it can be deleted.

Programmer response: Investigate any writer task abends.

Module: HSIZMON

HSIZ013I *1* USAGE MONITOR - UNRECOGNISED PROGRAM PARAMETER IGNORED

Explanation: An unrecognized program parameter was specified.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: Remove or correct the program parameter.

Programmer response: None.

Module: HSIZMON

HSIZ014I *1* USAGE MONITOR - COULD NOT OPEN FILE HSIZIN

Explanation: The HSIZIN file could not be opened by the Usage Monitor.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: Supply or correct the HSIZIN DD statement in the JCL.

Programmer response: None.

Module: HSIZMON

HSIZ015I *1* USAGE MONITOR - COULD NOT OPEN FILE HSIZMSG

Explanation: The HSIZMSG file could not be opened by the Usage Monitor.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: Supply or correct the HSIZMSG DD statement in the JCL.

Programmer response: None.

Module: HSIZMON

HSIZ016I *1* USAGE MONITOR TERMINATING - INVALID OR MISSING HSIZIN DATA

Explanation: At least one HSIZIN input statement was invalid, or input required to be present in the HSIZIN file was missing.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: Examine the HSIZMSG output report. Correct any invalid statements. Ensure a valid data set name prefix was specified.

Programmer response: None.

Module: HSIZMON

HSIZ017I *1* USAGE MONITOR TERMINATING - NOW WRITING CAPTURED DATA

Explanation: A STOP command has been encountered. The current repository contents are written before the Usage Monitor terminates.

In the message text:

1 current system identifier.

System action: The Usage Monitor starts a writer task and waits for its completion before terminating.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ018I *1* **USAGE MONITOR HAS NOW TERMINATED**

Explanation: The Usage Monitor has now freed resources and is about to terminate.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ019I *1* **USAGE MONITOR REPOSITORY FULL - NOW SWITCHING**

Explanation: The current Usage Monitor data collection repository is full.

In the message text:

1 current system identifier.

System action: A new repository is created and used for subsequent data collection. A writer task is initiated for the full repository.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ020I *1* **THE SPECIFIED NUMBER WAS TOO SMALL**

Explanation: The numeric value of a command subparameter was too small to be valid in the command context.

In the message text:

1 current system identifier.

System action: The command is discarded.

User response: Correct the numeric value and reissue the command.

Programmer response: None.

Module: HSIZMON

HSIZ021I *1* **THE SPECIFIED NUMBER WAS TOO LARGE**

Explanation: The numeric value of a command subparameter was too large to be valid in the command context.

In the message text:

1 current system identifier.

System action: The command is discarded.

User response: Correct the numeric value and reissue the command.

Programmer response: None.

Module: HSIZMON

HSIZ022I *1* **PASSIVE MODE SET FROM PROGRAM PARAMETER**

Explanation: PASSIVE was specified in the program parameter.

In the message text:

1 current system identifier.

System action: The Usage Monitor starts in passive mode unless overridden by input from the HSIZIN file.

User response: Set the Usage Monitor into collection mode to start data collection.

Programmer response: None.

Module: HSIZMON

HSIZ023I *1* **PROGRAM NAME MASK** *2* **NOT ADDED - ALREADY IN TABLE**

Explanation: A command to add a program name mask to a program mask table was issued, but the mask was already present in the table.

In the message text:

1 current system identifier.

2 program mask specified in command.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ024I *1* **PROGRAM NAME MASK** *2* **ADDED TO TABLE**

Explanation: A command to add a program name mask to a program mask table was issued, and the mask was added successfully.

In the message text:

1 current system identifier.

2 program mask specified in command.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ025I *1* **PROGRAM NAME MASK** *2* **NOT DELETED - NOT FOUND IN TABLE**

Explanation: A command to delete a program name mask from a program mask table was issued, but the mask was not present in the table.

In the message text:

1 current system identifier.

2 program mask specified in command.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ026I *1* **PROGRAM NAME MASK** *2* **DELETED FROM TABLE**

Explanation: A command to delete a program name mask to a program mask table was issued, and the mask was deleted successfully.

In the message text:

1 current system identifier.

2 program mask specified in command.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ027I **WARNING - ECSA APPEARS TO BE EXHAUSTED - INCREASE SIZE FOR NEXT IPL**

Explanation: The Usage Monitor has attempted to acquire storage from ECSA, but was given CSA storage by the system. This indicates that there is insufficient ECSA for the current workloads, and that it should be increased for the next IPL.

System action: Processing continues.

User response: Notify the system programmer.

Programmer response: Add around 50 to 100 megabytes to the ECSA size in the system IPL parameters. Check the capacity of the COMMON page data set.

Module: HSIZMON

HSIZ028I ***DANGER* - ECSA AND CSA APPEAR TO BE EXHAUSTED - INCREASE ECSA NEXT IPL**

Explanation: The Usage Monitor has attempted to acquire some common storage, but the requested amount was unavailable. This indicates that there is insufficient ECSA for the current workloads, and that it should be increased for the next IPL.

System action: Processing continues.

User response: Notify the system programmer.

Programmer response: Add around 50 to 100 megabytes to the ECSA size in the system IPL parameters. Close some applications using CSA. If necessary, commence orderly shutdown and re-IPL before the system crashes. Check the capacity of the COMMON page data set.

Module: HSIZMON

HSIZ029I *1* **THERE IS CURRENTLY NO EXCLUDE TABLE**

Explanation: A request was made to change or display the program name mask exclude table, but there is currently no exclude table.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: None required. The EXC command may be used to create a table.

Programmer response: None.

Module: HSIZMON

HSIZ030I *1* **USAGE MONITOR - NO DATA COLLECTED SO SKIPPING WRITE**

Explanation: Before a writer task was initiated to output the contents of a Usage Monitor repository, it was found that the repository contained no data, and that therefore data output processing could be omitted.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ031I *1* **INITIATING REPOSITORY SWITCH**

Explanation: A switch (SWI) command was issued and the requested action is being initiated.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ032I *1 2* **COMMAND UNKNOWN**

Explanation: A command was issued but was not recognized.

In the message text:

1 current system identifier.

2 name of the issued command.

System action: The command is ignored. Processing continues.

User response: If necessary, correct and reissue the command.

Programmer response: None.

Module: HSIZMON

HSIZ033I *1 2* **COMMAND PROCESSED**

Explanation: A command was issued and has been processed successfully.

In the message text:

1 current system identifier.

2 name of the issued command.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ034I *1 2* **COMMAND HAS INVALID OPERAND**

Explanation: A command was issued but an invalid operand was encountered.

In the message text:

1 current system identifier.

2 name of the issued command.

System action: The command is ignored. Processing continues.

User response: If necessary, correct and reissue the command.

Programmer response: None.

Module: HSIZMON

HSIZ035I *1 2* **COMMAND FAILED**

Explanation: A command was issued but insufficient resources were available to execute it successfully.

In the message text:

1 current system identifier.

2 name of the issued command.

System action: The command is ignored. Processing continues.

User response: Try again after more resources become available.

Programmer response: None.

Module: HSIZMON

HSIZ036I *1 2* **COMMAND CAUSED NO CHANGE**

Explanation: A command was issued but the state to be set by the command was found to already exist.

In the message text:

1 current system identifier.

2 name of the issued command.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ037I *1 2* **COMMAND REJECTED**

Explanation: A recognized command was issued at a time when the Usage Monitor is unable to process the command.

In the message text:

1 current system identifier.

2 name of the issued command.

System action: The command is ignored. Processing continues.

User response: Try again after the Usage Monitor has freed the resources.

Programmer response: None.

Module: HSIZMON

HSIZ038I *1* **CURRENT USAGE MONITOR PROGRAM EXCLUDE LIST:**

Explanation: A D-X command was issued to display the program name exclude table contents. The active entries are shown after this message.

In the message text:

1 current system identifier.

System action: The data is displayed and processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ039I *1* **REPOSITORY SWITCH HAS BEEN QUEUED**

Explanation: A repository switch was triggered by a SWI or STOP command, or by the current repository becoming full, but a writer task is already active. This message is followed by message HSIZ040I which shows the creation timestamp of the active writer task.

In the message text:

1 current system identifier.

System action: Data collection is suspended. Wait for the current writer task to complete whereupon a new writer task is created, and a new repository is created, and data collection is resumed.

User response: Check that there are sufficient resources to dispatch the Usage Monitor address space. Check that there are no serialization problems with system components such as device allocation which could be inhibiting writer task processing.

Programmer response: None.

Module: HSIZMON

HSIZ040I *1* **WAITING FOR WRITER TASK ATTACHED** *2*

Explanation: A repository switch was triggered by a SWI or STOP command, or by the current repository becoming full, but a writer task is already active. This message follows message HSIZ039I and shows the creation timestamp of the active writer task.

In the message text:

1 current system identifier.

2 Time stamp of write task.

System action: Data collection is suspended. Wait for the current writer task to complete whereupon a new writer task is created, and a new repository is created, and data collection is resumed.

User response: Check that there are sufficient resources to dispatch the Usage Monitor address space. Check that there are no serialization problems with system components such as device allocation which could be inhibiting writer task processing.

Programmer response: None.

Module: HSIZMON

HSIZ041I *1* **CURRENT USAGE MONITOR OUTPUT DYNALLOC PARMS:**

Explanation: A D-A command was issued to display the current output dynamic allocation parameters, which are shown after this message.

In the message text:

1 current system identifier.

System action: The data is displayed and processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ042I **CURRENT USAGE MONITOR OUTPUT SYSTEM ID IS** *"1"*

Explanation: A D-I command was issued to display the current system identifier which is to be contained in output header records.

In the message text:

1 System ID of current system.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ043I *1* **WARNING - DATA DISCARDED DUE TO (E)CSA STORAGE LIMIT**

Explanation: The Usage Monitor has detected for the first time in the life of the repository or since a CSA limit change that program usage event data has been discarded due to the CSA/ECSA storage usage limit being reached. This limit was set with the CSA command.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: Adjust the Usage Monitor CSA limit as appropriate for the particular system. Ensure that the ECSA size has been generously defined for the system, and that the common page data set size is adequate. Ensure that the Usage Monitor address space is running at a higher priority than all CPU-bound workloads. Generally, monitors need to run at a higher priority than the workloads being monitored.

Programmer response: None.

Module: HSIZMON

HSIZ044I *1* **SWITCH-AND-WRITE TIME-OF-DAY IS SET TO** *2*:*3*

Explanation: A D-T command was issued to display the switch-and-write time-of-day setting for this system. hh:mm switch time-of-day.

In the message text:

1 current system identifier.

2 HH - Hour of the day.

3 MM - minute of the hour.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ045I *1* **CREATED REPOSITORY** *2*-*3*

Explanation: A repository was created to hold collected program usage data.

In the message text:

1 current system identifier.

2 space token of the repository data space.

3 ALET of the repository data space.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ046I *1* **DELETED REPOSITORY** *2*-*3* *4* **ENTRIES CACHED**

Explanation: A repository which was no longer needed was deleted.

In the message text:

1 current system identifier.

2 space token of the repository data space.

3 ALET of the repository data space.

4 number of entries cached to reduce overhead.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ047I *1* **USAGE MONITOR - ATTACHING WRITER SEQ-NO-***2*

Explanation: A writer task is being attached to write out repository contents. The writer task sequence number is also reported. The first writer task to run after the Usage Monitor starts has a sequence number of 1.

In the message text:

1 current system identifier.

2 sequence number of writer task this run.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ048I *1* **USAGE MONITOR - IDENTIFY FAILED HEX RC=***2*

Explanation: The Usage Monitor executed an IDENTIFY macro which failed.

In the message text:

1 current system identifier.

2 hexadecimal return code of the IDENTIFY macro.

System action: Processing terminates.

User response: Notify the system programmer.

Programmer response: Investigate why an IDENTIFY macro would fail with that return code.

Module: HSIZMON

HSIZ049I *1* **DATA SET NAME MASK NOT DEACTIVATED, NOT FOUND IN LIST**

Explanation: A command to delete a data set name mask from a data set name mask list was issued, but the mask was not present in the list.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ050I *1* **DATA SET NAME MASK** *2* **LIST** *3*

Explanation: A D-D command was issued to display the data set name mask include and exclude lists. These header and trailer lines mark the start and end of the lists.

In the message text:

1 current system identifier.

2 INCLUDE or EXCLUDE.

3 START or END.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ051I *1* **(E)CSA QUEUING STORAGE LIMIT:** *2*

Explanation: Either a CSA command was issued to change the limit setting, or a D-S command was issued. The CSA queuing storage limit can be used to limit the quantity of CSA to be used to hold program usage data elements queued for storing into the data space repository. When this limit is reached further data is discarded. A count of discarded elements is maintained an reported at termination. A limit of zero means the usage monitor never tries to limit CSA storage usage.

In the message text:

1 current system identifier.

2 NO LIMIT or kilobyte limit.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ052I *1* **THE CACHE TRIGGER EVENT COUNT IS** *2*

Explanation: Either a TRG command was issued to change the cache trigger event count or a D-S command was issued. When a job uses the same program a number of times, the repository entry may be cached if the number has reached the cache trigger event count. The updating of cached entries is a synchronous process which does not use common storage. A limited number of cache entries are available. The cache is cleared when the repository is switched.

In the message text:

1 current system identifier.

2 event count required to cause caching.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ053I *1* **MONITORING UNIX PROGRAMS?** *2*

Explanation: Either a USS command was issued to change the UNIX program monitoring status or a D-S command was issued. When the answer is YES the usage of programs fetched from UNIX files is monitored. When the answer is NO only the usage of programs from PDS and PDSE libraries is monitored.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ054I *1* **MONITORING LINK PACK AREA PROGRAMS?** *2*

Explanation: Either an LPA command was issued to change the LPA program monitoring status or a D-S command was issued. When the answer is YES the usage of programs residing in the Link Pack Area is monitored. When answer is NO only the usage of programs loaded into address space regions (and sometimes into CSA) is monitored.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ055I *1* **FLAG SYSPLEX-WIDE COMMON SOFTWARE?** *2*

Explanation: Either a PLX command was issued to change the sysplex status flag or a D-S command was issued. This flag setting is copied to the header record every time a data set is written by the Usage Monitor. This flag can be used to determine how usage data relates to Inquisitor data. In particular, if more than one Operating System image entirely shares a common DASD subsystem, a YES value can enable the usage data of more than one system to relate to a single Inquisitor data collection.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ056I *1* **PREFER VOLUME SYMBOL OVER SERIAL?** *2*

Explanation: Either a SYM command was issued to change the volume symbol status or a D-S command was issued. When the answer is YES the volume serial number field of data records contains six asterisks instead of the IPL volume serial ("system residence volume") and symbol names for other volumes if a system static symbol can be found with a value matching the captured volume serial number. Otherwise the volume serial number is written as normal. When the answer is NO the captured volume serial number is always output. A YES setting may be useful to improve data matching when system software platform volume switches take place.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ057I *1* **WILL WRITER TASK SORT THE DATA?** *2*

Explanation: Either a SRT command was issued to change the SORT status or a D-S command was issued. When the answer is YES the writer task attaches the site's SORT utility to sort the data before it is written to the output data set. When the answer is NO data is written in the (hashed) order it is stored in within the repository. Usage import processing time may be significantly reduced when the data is sorted. The Usage Monitor address space consumes more resources if the writer task is to sort the data. Local customization of SORT settings and/or Usage Monitor JCL may be required to ensure that sufficient SORTWORK space is available when sorting is to be performed.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ058I *1* **FILE HSIZIN IS NOT ALLOCATED - CANNOT PERFORM REFRESH**

Explanation: A REF command was issued to refresh settings from commands in the HSIZIN file, but the HSIZIN file had been freed, and was no longer allocated to the Usage Monitor.

In the message text:

1 current system identifier.

System action: The refresh operation is suppressed and processing continues.

User response: Ensure FREE=CLOSE is not specified in the HSIZIN JCL DD statement. Recycle the Usage Monitor to refresh the settings if necessary.

Programmer response: None.

Module: HSIZMON

HSIZ059I *1* **REFRESH PERFORMED WITH NO ERRORS**

Explanation: A REF command was issued to refresh settings from commands in the HSIZIN file. All commands in the HSIZIN file were completed successfully.

In the message text:

1 current system identifier.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ060I *1* **REFRESH PERFORMED BUT ERROR(S) FOUND**

Explanation: A REF command was issued to refresh settings from commands in the HSIZIN file. At least one command in the HSIZIN file resulted in an error.

In the message text:

1 current system identifier.

System action: Processing terminates.

User response: Examine the output in the HSIZMSG file to determine the problem(s).

Programmer response: None.

Module: HSIZMON

HSIZ062I *1* **MAXCAD=***2* **IS PROBABLY TOO SMALL**

Explanation: A DSPSERV CREATE macro issued a return code of 12. This is usually caused by the maximum number of SCOPE=COMMON data spaces already existing, so that no more can be created. To increase this maximum specify a larger value for MAXCAD in the system parameter library for the next IPL.

In the message text:

1 current system identifier.

2 current value of MAXCAD.

System action: Processing terminates.

User response: Restart the Usage Monitor after a SCOPE= COMMON data space has been deleted.

Programmer response: Allow a greater number of concurrent SCOPE=COMMON data spaces by increasing MAXCAD in PARMLIB.

Module: HSIZMON

HSIZ063I *1* **COLLECTING ″UNKNOWN″ EVENTS?** *2*

Explanation: Either a UNK command was issued or a D-S command was issued. When the answer is YES this message indicates that the Usage Monitor logs events with incomplete data which would not normally be collected. Data base content is not affected.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ064I *1* **WILL WRITER TASK COMPRESS THE DATA?** *2*

Explanation: Either a ZIP command was issued to change the output compression setting or a D-S command was issued. When the answer is YES the writer task writes compressed data to reduce I/O volumes.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ065I *1* **WILL WRITER TASK CORRECT LINKLIST DSN?** *2*

Explanation: Either an LLC command was issued or a D-S command was issued. When the answer is YES the writer task will perform a BLDL for programs known to have been fetched from the link list, and each output record for such programs will be altered to reflect the link list data set name that the writer task found the program in.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ066I *1 2* **IDLE ELEMENT(S) ″LOST″ DUE TO ZERO POINTER**

Explanation: The Usage Monitor was terminating normally when a storage accounting discrepancy was discovered. The storage for the idle element chain was being freed when it was found to be terminated by a zero pointer before the expected number of elements had been processed. The most probable cause is a storage overlay. This may or may not represent a Usage Monitor logic error. The size of common storage which may be unusable until the next IPL can be calculated by multiplying the element count by the size of an element.

In the message text:

1 current system identifier.

2 the number of elements being reported.

System action: Termination continues.

User response: Determine if the size of the potential loss of common storage is likely to impact upon system stability, and take the appropriate action. Ensure that all appropriate maintenance has been applied.

Programmer response: None.

Module: HSIZMON

HSIZ067I *1* **SAVE ELEMENTS ON THE IDLE CHAIN?** *2*

Explanation: Either an IDL command was issued or a D-S command was issued. When the answer is YES the Usage Monitor will place processed work elements on a chain for idle elements instead of freeing the storage. When an address space needs an element to record a program usage event, one from the idle chain will be used in preference to acquiring more storage. Use of the idle chain can reduce GETMAIN/FREEMAIN processing and therefore improve overall efficiency.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None required.

Programmer response: None.

Module: HSIZMON

HSIZ068I *1* **GET JOB ACCOUNT NOW?** *2*

Explanation: A D-S command was issued. When the answer is YES job account data is currently being collected as program usage events are recorded. When the answer is NO job account data is not being collected currently.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ069I *1* **GET JOB ACCOUNT LATER?** *2*

Explanation: Either a JAC command was issued or a D-S command was issued. When the answer is YES job account data will be collected after the next Usage Monitor collection repository switch. If the answer is NO job account data will not be collected from that time onwards.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ070I *1* **GET REGISTERED PRODUCT DATA NOW?** *2*

Explanation: A D-S command was issued. When the answer is YES registered software product data from SMF is currently being collected by the Usage Monitor. When the answer is NO then this SMF data is not being currently collected.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ071I *1* **GET REGISTERED PRODUCT DATA LATER?** *2*

Explanation: Either a PRS command was issued or a D-S command was issued. When the answer is YES registered software product data from SMF will be collected after the next Usage Monitor collection repository switch. When the answer is NO this SMF data will not be collected after the next switch.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ072I *1* **GET DYNAMIC CAPACITY DATA NOW?** *2*

Explanation: A D-S command was issued. When the answer is YES, hardware capacity information is currently being collected by the Usage Monitor. When the answer is NO, hardware capacity information is not being currently collected.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ073I *1* **GET DYNAMIC CAPACITY DATA LATER?** *2*

Explanation: Either a CAP command was issued or a D-S command was issued. When the answer is YES the Usage Monitor will collect hardware capacity information after the next Usage Monitor collection repository switch. When the answer is NO the hardware capacity information will not be collected after the next switch.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ074I *1* **OUTPUT NAMES OF USERS?** *2*

Explanation: Either a UNM command was issued or a D-S command was issued. When the answer is YES collected user names will be included in the data output by the Usage Monitor writer task. When the answer is NO user names will not be written to the output data set.

In the message text:

1 current system identifier.

2 YES or NO.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ080I *1 2*

Explanation: Displays the dataset name mask for a D-D command.

In the message text:

1 current system identifier.

2 Data set name.

System action: Processing continues.

User response: None.

Programmer response: None.

Module: HSIZMON

HSIZ201I **DYNALLOC FAILURE RC=***1* **ERROR=***2* **INFO=***3* **DSN=***4*

Explanation: The writer task could not dynamically allocate a new output data set.

In the message text:

1

2 dynamic allocation return code (DARC).

3 dynamic allocation information code.

4 name of the data set being allocated.

System action: Processing of the repository is terminated, and the data lost.

User response: Correct the cause of the allocation failure. If necessary, use the DSN, PRI, SEC and UNT commands to customize the allocation request for your installation. Note: The meanings of most DARC values are usually available in Appendix A of the ISPF Tutorial.

Programmer response: None.

Module: HSIZ0203

HSIZ202I **USAGE MONITOR - COMPRESSION SUBROUTINE ERROR**

Explanation: While processing repository data the compression subroutine encountered an error. The error message from the compression subroutine immediately follows this message.

System action: Processing of the repository is terminated, and the data lost.

User response: Correct the error described in the message from the compression subroutine. If necessary, contact IBM support.

Programmer response: None.

Module: HSIZ0203

HSIZ203I **USAGE MONITOR - SORT FAILED - RC=***3*

Explanation: While sorting repository data the SORT task ended with a non-zero condition code which is taken to mean that the sort was not successful. This message is followed by message HSIZ205I.

In the message text:

3 decimal return code of the sort subtask.

System action: The output data set is closed, and the writing of unsorted data to the same data set is attempted.

User response: Consult the documentation of the SORT utility. The contents of the SORT report file (DDNAME=SYSOUT) may be helpful.

Programmer response: None.

Module: HSIZ0203

HSIZ204I **USAGE MONITOR - SORT ABENDED** *-1*

Explanation: While sorting repository data the SORT task ended abnormally. This message is followed by message HSIZ205I.

In the message text:

1 the abend code of the sort subtask.

System action: The output data set is closed, and the writing of unsorted data to the same data set is attempted.

User response: Investigate why such an abend could occur. The contents of the SORT report file (DDNAME=SYSOUT) may be helpful.

Programmer response: None.

Module: HSIZ0203

HSIZ205I USAGE MONITOR - UNSORTED DATA WILL BE WRITTEN

Explanation: The sorting of output data has failed so the data is now written unsorted.

System action: The message is preceded by either HSIZ203I or HSIZ204I. After the SORT task ended the output data set has been closed and reopened. Repository data is about to be written to the output data set.

User response: None required, other than investigating why the sort failed.

Programmer response: None.

Module: HSIZ0203

HSIZ206I *1*

Explanation: The HSISHRNK compression routine issued an error message which is displayed.

In the message text:

1 Error message from HSISHRNK

System action: The message is preceded by message HSIZ202I.

User response: Examine the message for further information.

Programmer response: None.

Module: HSIZ0203

HSIZ301I DESERV FUNC=EXIT RC=*1* REASON=*2*

Explanation: DESERV FUNC=EXIT issued a non-zero return code.

In the message text:

1 Return code from DESERV.

2 Reason code from DESERV.

System action: The DESERV exit is not installed.

User response: Notify the system programmer.

Programmer response: Research the DESERV feedback to determine why the exit could not be installed.

Module: HSIZ0303

HSIZ302I CSVDYNEX ADD (*1*) RC=*2* REASON=*3*

Explanation: CSVDYNEX ADD issued a non-zero return code.

In the message text:

1 xxxxx

2 Return code from CSVDYNEX.

3 Reason code from CSVDYNEX.

System action: The SMF exit is not installed.

User response: Notify the system programmer.

Programmer response: Research the CSVDYNEX feedback to determine why the exit could not be installed.

Module: HSIZ0303

HSIZ303I ATTRIBUTE MISMATCH - *1* NOT INSTALLED

Explanation: The examined SVC table entry did not have the expected attributes.

In the message text:

1 Module name.

System action: The SVC intercept is not installed.

User response: Notify the system programmer.

Programmer response: Contact IBM support.

Module: HSIZ0303

HSIZ306I BAD *1* ENTRY PGM=*2* JOB=*3* USER=*4* ID=*5* DATE=*6* REJECTED

Explanation: An invalid work element has been detected and some of its contents are displayed.

In the message text:

1 Status name.

2 Program name.

3 Job name.

4 User name.

5 ID name.

6 Date.

System action: Attempted to dump some data to HSIZSNAP if the file is allocated, and then tries to free the work element without processing its contents.

User response: Notify the system programmer.

Programmer response: The problem is indicative of a storage overlay. Contact IBM support.

Module: HSIZ3060

HSIC - Globalization messages

Return codes

0	No errors encountered. All requests processed successfully.
16	Unrecoverable error. No requests processed. SYSIN or HSIPZIP or INQSOUT File cannot be used, or unsupported operating system.

Message suffix codes

Suffix	Meaning	Raises Minimum Cond Code to
I	Information Message	0
W	Warning Message	4
E	Error Message	8
S	Severe Error Message	12
U	Unrecoverable Error Message	16

Message texts and explanations

All numeric completion codes of system services reported in these messages are in hexadecimal unless otherwise stated.

HSIC002E A message is missing from the internal repository

Explanation: A message is missing from the internal message repository. When the default language is not English, it could simply mean that no translation of the given message exists. If the default language is English, that would indicate an error in the given application.

System action: The application would normally continue ignoring the given message number , but the specific action depends on the code attempting to issue the message which could also terminate the application.

User response: Contact your IBM representative.

HSIC003U The internal message repository is corrupted.

Explanation: When attempting to issue a message, the internal message repository layout did not follow the expected format.

System action: The application terminates.

User response: Contact your IBM representative.

HSIC020E *application-name* **encountered errors. Error code =** *errorcode*

Explanation: The Application has encountered errors during processing. This is a general message on completion indicating that an error has occurred.

System action: Completes with given error code .

User response: Refer to additional message, or to the section "Return codes" on page 184, and to the log for more details on the specific error. If necessary, contact your IBM representative.

HSIC021S *application-name* **encountered fatal errors. Error code =** *error-code*

Explanation: The Application has encountered fatal errors during processing.

System action: Terminates with given error code

User response: Refer to additional message, or to the section "Return codes" on page 184, and to the log for more details on the specific error. If necessary, contact your IBM representative.

HSIC023E Inquisitor Import error occurred in opening: *filename*

Explanation: The Inquisitor import could not open the given file.

System action: Terminates without processing any records.

User response: Check that the file exists, and if it does, check for any additional log message identifying the error. If necessary, contact your IBM representative.

HSIC024E Inquisitor Import input file is in error.

It looks like a usage data file

Explanation: The inquisitor import has encountered an invalid input file.

System action: Terminates without processing any records.

User response: Check that the input file is a valid file. If necessary, contact your IBM representative.

HSIC025E Inquisitor Import input file is in error. It looks like a hardware data file

Explanation: The Inquisitor Import has encountered an invalid input file.

System action: Terminates without processing any records.

User response: Check that the input file is a valid file. If necessary, contact your IBM representative.

HSIC026E Inquisitor Import detected that table *tablename* **is missing or invalid.**

Explanation: The expected table is missing from the database or has invalid format. This suggests a mismatch between the database and this version of the product.

System action: Terminates without processing any records.

User response: Check for a version mismatch between the database and the version of the product. If necessary, contact your IBM representative.

HSIC027S Inquisitor Import table *tablename* **is missing a column.**

Explanation: The given table is missing an expected column. This suggests a mismatch between the database and this version of the product.

System action: The application terminates without processing any records.

User response: Check for a version mismatch between the database and the version of the product. If necessary, contact your IBM representative.

HSIC028S Inquisitor Import table *tablename* **appears to be an old version.**

Explanation: The given table in the database does not have the expected format.

System action: The application terminates without processing any records.

User response: Check for a version mismatch between the database and the version of the product. If necessary, contact your IBM representative.

HSIC029S Inquisitor Import error when writing to table *tablename*

Explanation: An SQL error occurred when attempting to write to the given table.

System action: The application terminates.

User response: Check the log for additional details about the given error. If necessary, contact your IBM representative.

HSIC030S The Inquisitor Import did not find a valid system header record in the input file

Explanation: The input file does not follow the expected format.

System action: The application terminates.

User response: Check that the correct input file is supplied, and that there is no version mismatch. If necessary, contact your IBM representative.

HSIC034S Error reading Repository TPARAM table

Explanation: An error occurred while reading the TPARAM Repository table.

System action: The application terminates.

User response: Check the log for any additional messages indicating the cause of the error. If necessary, contact your IBM representative.

HSIC035E The Repository is in use by the *application-name*

Explanation: The application cannot run because the Repository is already in use by another application. Wait until *application-name* completes before running the current application. If the Repository is not in use by *application-name*, then the cause could be that it was previously run, but did not run to completion. To correct the problem, either rerun the *application-name* identified in this message, or alternatively, run the HSISTPRM supplied job to reset FVALUE to 0 where FKEY = PROCRUN in the TPARAM table.

System action: The application terminates.

User response: Check the application is not already in use, before running this application.

HSIC036E Syntax error scanning TPARAMS file on line *linenumber*

Explanation: The TPARAM file does not conform to the required syntax on the given line.

System action: The specified option or value is ignored, and its default value is used where applicable.

User response: Check that valid options/values are supplied as specified in the documentation of the application that you are running.

HSIC037E Schema *schemavalue* is too long in param *param*

Explanation: A schema id that is too long has been specified.

System action: The application terminates.

User response: Check that the schema id does not exceed 8 characters in length.

HSIC038E Unbalanced quote for value: *value* in param: *param*

Explanation: A starting quote was found for the given parameter that has no matching end quote.

System action: The application terminates.

User response: Check that the given parameter has matching quotes

HSIC039E Illegal character in value*:value* of param:*param*

Explanation: An invalid character was found in the given value.

System action: The application terminates.

User response: Check that the given parameter value is valid for its type.

HSIC040E Reserved word: *reservedword* in param: *param*

Explanation: A reserved word or system value schema ID was chosen as a parameter value.

System action: The application terminates.

User response: Specify a different parameter value

HSIC041W value:*value* in param:*param* is not a recommended schema ID

Explanation: The value is not recommended because of possible conflicts with existing values.

System action: The application continues.

User response: Please choose a different value to avoid any conflicts

HSIC042E TPARAM file: param*:param* has an invalid proposed value: *value*

Explanation: The parameter cannot be set to the given value, because the value is not valid..

System action: The value is ignored, and the application continues.

User response: Please choose a valid value as per the documentation of the given application

HSIC043E The application has failed to open the TPARAM file. Error: *errordescription*

Explanation: The application could not open the TPARAM file. The error description contains more details regarding the reason for the error.

System action: The application terminates.

User response: Check that the TPARAM file exists and is valid.

HSIC045E String *string* cannot exceed *numberchars* in length

Explanation: A parameter length limit has been exceeded.

System action: The application terminates.

User response: Ensure that the specified parameter length is not exceeded.

HSIC050E The *program-name* program has detected an invalid date parameter

Explanation: A date parameter was found to be invalid.

System action: The application terminates.

User response: Ensure that the date format is valid, and start dates do not overlap end dates.

HSIC051S Error adding record

Explanation: An SQL error occurred when adding a record to a table.

System action: The application terminates.

User response: Check the log for additional information about the error. If necessary contact your IBM representative.

HSIC052S Error updating record

Explanation: An SQL error occurred when updating a record in a table.

System action: The application terminates.

User response: Check the log for additional information about the error. If necessary contact your IBM representative.

HSIC053S Error deleting record

Explanation: An SQL error occurred when deleting a record from a table.

System action: The application terminates.

User response: Check the log for additional information about the error. If necessary contact your IBM representative.

HSIC054E **Usage Summary detected an invalid SUMBY value**

Explanation: The Usage Summary detected an invalid SUMBY value.

System action: The specified value is ignored. The application continues using the default SUMBY value.

User response: Refer to the documentation of the Usage Summary parameter for valid SUMBY values.

HSIC055S **Table initialization failure during Repository Merge**

Explanation: At least one table initialization failed when merging repositories.

System action: The application terminates.

User response: Check the log for any additional details about this error. If necessary, contact your IBM representative.

HSIC056S **Some table destination fields are smaller than source**

Explanation: Some fields in the target repository are not large enough to fit the contents of fields in the source repository.

System action: The application terminates, and the repositories are not merged.

User response: Check that the destination repository is not an older version than the source repository. If necessary, you can recreate the destination repository using the latest version of the product. If the problem persists, contact your IBM representative.

HSIC057E **A value for parameter:** *parameter-name* **must be specified**

Explanation: A mandatory parameter for this application has not been specified.

System action: The application terminates during the syntax checking of input parameters.

User response: Ensure that a value for the given parameter is specified. Refer to the documentation of the failing application for an explanation of the given parameter and/or valid parameter values.

Return codes

6016	**Input text file open error**
6060	**Input Parameter error**
6061	**Database open error**
6062	**Database commit error**
6063	**Error reading repository TPARAM table**
6065	**Repository is in use**
6203	**Inquisitor Import table open fail**
6204	**MVS system header record not found in input file**
6205	**Unix System Services header record not found in input file**
6206	**No system header record found in input file.**
6208	**Error writing to TPARAM table**
6209	**Error opening input file**
6211	**Fatal error writing system record**
6212	**Fatal error writing library record**
6213	**Fatal error writing module record**
6218	**Input file looks like a usage data file**
6219	**Input file looks like a hardware data file**
6220	**Index missing error**
6221	**Vendor product version table processing error**
6222	**Tagged module key table processing error**
6226	**Repository must be enabled for Unix System Services, when the REPLACE option is in effect.**

6237	Inquisitor Import table does not exist or is a missing a column
6238	Inquisitor Import table does not exist
6239	Inquisitor Import table appears to be an old version.
6240	Error updating fGPassLibID record
6241	Error deleting empty libraries
6244	Error assigning package information to TMODULE records
6260	Nothing to import, as no module records were found in IQ file.
6400	Knowledge Base type is incorrect
6402	Failure in initializing IQ tables
6403	IQ TMODULE open error
6404	IQ TMODULE index error
6405	IQ database is empty
6409	TDECISION table open error
6413	Error creating scorecard tables for Match Engine
6428	Local KB TRULES table open error
6434	Failure to open archive file
6435	Error creating index
6436	Error setting current index
6437	Search KB phase error
6438	Volume serial library phase error
6439	Inter Library phase error
6440	Rules processing phase error
6444	LPA phase error
6448	Error while clearing LMOD count
6449	TDECISION Table is missing FDECRPTION and/or FCATEGORY fields
6450	GKB TPRODUCT record seek error
6451	LKB TPRODUCT record seek error
6452	TDECISION record edit error
6453	KB TVERSION record access error
6454	KB TPRODUCT record access error
6455	KB TVENDOR record access error
6600	Match Engine tables TDECISION and/or TMIGREPORT are missing
6619	Error opening TPACKAGE table
6620	Repository table initialization failed
6621	Failure opening IQ table
6622	Unable to access GKB TVERSION table
6623	IQ TMODULE table is empty
6624	Predecessor inventory ID key does not exist
6625	Repository is not enabled for Unix System Services
6626	Repository must be enabled for Unix System Services, when the REPLACE option is in effect.

6627	**SYSPLEX ID mismatch in inventory record**
6628	**SMFID mismatch in inventory record**
6629	**Inventory ID key of zero is not valid**
6630	**Error in deleting library record**
6632	**Error transferring TLIBRARY information from IQ to Repository**
6633	**Error accessing TINVCTL table**
6634	**Mismatch found between the TINVCTL record flag and the REPLACE option.**
6635	**Error updating FMODCNT field in TLIBRARY and TPOVLIB tables**
6636	**Product version key error**
6637	**Module key error**
6639	**Error updating FINVID18 fields in TUIMPORTCTRL table**
6640	**Error updating FINVID field in TINVREG table**
6641	**Error updating FINVID field in TINVREG table**
6642	**Error updating summary tables**
6643	**Error querying table in FMODID order**
6645	**Error marking TLIBRARY, TMODULETPOVLIB and TPOVINV records as deleted.**
6647	**Repository type does not match IQ type**
6648	**When using a Continuous Inventory, an Inventory Name must be specified**
6800	**At least one repository failed during initialization**
6802	**No matching LPAR found in table**
6803	**Primary Inventory ID set to 0 for LPAR**
6804	**Error trying to find FMODID or FLIBID**
6805	**Inventory ID does not exist**
6806	**Unable to find or create TLPAR record for LPAR**
6807	**Error trying to find or create Job or User entry**
6808	**Error writing MTD record**
6809	**Error updating summary tables**
6810	**Error adding TUSELIBRARY record**
6811	**TLIBRARY update error**
6812	**Summary table error**
6813	**Error reading import control record**
6814	**User initiated stop**
7000	**At least one table failed initialization**
7002	**Invalid usage summary parameters**
7003	**Invalid month in usage summary parameter**
7004	**Date order error**
7005	**TMODULE record seek error**
7011	**Error inserting record into TMODULE table**

7013	**TJOBDATA record seek error**
7014	**TJOBDATA record add error**
7015	**TUSERDATA record seek error**
7016	**TUSERDATA record add error**
7017	**TUSEMTD record seek error**
7018	**TUSEMTD record add error**
7019	**TUSEMTD record edit error**
7020	**TUSEMTD record delete error**
7021	**TPOVINV record seek error**
7022	**TPERIODS record seek error**
7023	**TPERIODS record add error**
7024	**TPERIODS record edit error**
7025	**TUSEPOVLIB record seek error**
7026	**TUSEPOVLIB record add error**
7027	**TUSEPOVLIB record edit error**
7028	**TUSEPOV record seek error**
7029	**TUSEPOV record add error**
7030	**TUSEPOV record edit error**
7034	**TUSEMTD critical failure**
7035	**TUSEMTD error updating record with zero FMTDID**
7036	**TVERSION record seek error**
7037	**TUSEPO record seek error**
7038	**TUSEPO record seek error**
7039	**TUSEPO record edit error**
7040	**TUSEPO record delete error**
7043	**TMODULE record edit error**
7044	**TUSEPOVLIB record delete error**
7045	**TUSEPOV record delete error**
7046	**TPERIODS record delete error**
7051	**TUSELIB record delete error**
7052	**IDS_USUM_TUSELIB_ AUTONUM_ERROR**
7055	**TLPAR record edit error**
7056	**TUSELIB record seek error**
7057	**TUSELIB record add error**
7058	**TPOVLIB record seek error**
7060	**TLPAR record seek error**
7061	**Join record seek error**
7062	**TLIBRARY record edit error**
7063	**TLIBRARY record seek error**
7065	**Invalid SUMBY value**
7066	**Date formatting error**
7067	**Usage Summary schema is empty**
7068	**PRODUCT_USE delete error**
7069	**PRODUCT_USE_DETAIL delete error**

7201	**Inventory to be deleted does not exist in repository**
7203	**TLIBRARY record delete failure**
7204	**TPOVINV record delete failure**
7205	**TPERIODS record delete failure**
7206	**TLPAR record delete failure**
7207	**TUIMPCTRL record delete failure**
7208	**Failure updating Delete Inventory ID record.**
7209	**Failure deleting TINVCTL records of deleted inventory**
7210	**Error scanning product version**
7211	**Error reassigning predecessor links in successor InvCTL records**
7600	**Table initialization failure**
7601	**Destination repository column size failure**
7602	**TINVCTL record seek error**
7603	**TINVCTL record edit error**
7604	**TINVCTL record add error**
7605	**TINVCTL record delete error**
7606	**TLIBRARY record seek error**
7607	**TLIBRARY record edit error**
7608	**TLIBRARY record add error**
7609	**TLIBRARY record delete error**
7610	**Transfer product version join seek error**
7611	**TPOVLIB record seek error**
7612	**TPOVLIB record edit error**
7613	**TPOVLIB record add error**
7614	**TPOVLIB record delete error**
7615	**TPOVINV record seek error**
7616	**TPOVINV record edit error**
7617	**TPOVINV record add error**
7618	**TPOVINV record delete error**
7619	**Table TINVPOV failed in initialization**
7620	**TVERSION record seek error**
7621	**TVERSION record edit error**
7622	**TVERSION record add error**
7623	**TVERSION record delete error**
7624	**Table TVERSION open failed**
7625	**TPRODUCT record seek error**
7626	**TPRODUCT record edit error**
7627	**TPRODUCT record add error**
7628	**TPRODUCT record delete error**
7629	**TPRODUCT open error**
7630	**TVENDOR record seek error**
7631	**TVENDOR record edit error**

7632	**TVENDOR record add error**
7633	**TVENDOR record delete error**
7634	**TVENDOR open error**
7635	**TMODULE record seek error**
7636	**TMODULE record edit error**
7637	**TMODULE record add error**
7638	**TMODULE record delete error**
7639	**TREGCLASS record seek error**
7640	**TREGCLASS record edit error**
7641	**TREGCLASS record add error**
7642	**TREGCLASS record delete error**
7643	**TREGION record seek error**
7644	**TREGION record edit error**
7645	**TREGION record add error**
7646	**TREGION record delete error**
7647	**TREGLEAF record seek error**
7648	**TREGLEAF record edit error**
7649	**TREGLEAF record add error**
7650	**TREGLEAF record delete error**
7651	**TINVREG record seek error**
7652	**TINVREG record edit error**
7653	**TINVREG record add error**
7654	**TINVREG record delete error**
7655	**TJOBDATA record seek error**
7656	**TJOBDATA record edit error**
7657	**TJOBDATA record add error**
7658	**TJOBDATA record delete error**
7659	**TUSERDATA record seek error**
7660	**TUSERDATA record edit error**
7661	**TUSERDATA record add error**
7662	**TUSERDATA record delete error**
7663	**TLPAR record seek error**
7664	**TLPAR record edit error**
7665	**TLPAR record add error**
7666	**TLPAR record delete error**
7667	**TUSEMTD record seek error**
7668	**TUSEMTD record edit error**
7669	**TUSEMTD record add error**
7670	**TUSEMTD record delete error**
7671	**TUSELIB record seek error**
7672	**TUSELIB record edit error**
7673	**TUSELIB record add error**
7674	**TUSELIB record delete error**
7675	**TPERIODS record seek error**

7676	**TPERIODS record edit error**
7677	**TPERIODS record add error**
7678	**TPERIODS record delete error**
7679	**TUSEPOVLIB record seek error**
7680	**TUSEPOVLIB record edit error**
7681	**TUSEPOVLIB record add error**
7682	**TUSEPOVLIB record delete error**
7683	**TUSEPOVLIB open error**
7684	**TUSEPOV record seek error**
7685	**TUSEPOV record edit error**
7686	**TUSEPOV record add error**
7687	**TUSEPOV record delete error**
7688	**TUSEPOV open error**
7689	**TUSEPO record seek error**
7690	**IDS_MRGE_TUSEPO_EDIT_ERROR**
7691	**TUSEPO record add error**
7692	**TUSEPO record delete error**
7693	**TUSEPO open error**
7694	**TUIMPORTCTRL record seek error**
7695	**TUIMPORTCTRL record edit error**
7696	**TUIMPORTCTRL record add error 7697**
7697	**TUIMPORTCTRL record delete error**
7698	**Source and destination repositories are not the same type**
7699	**Source and/or Destination Repositories are not the correct category database**

Appendix B. Repository table layouts

Table 17. TINVCTL

Column Name	Column Type	Column Length	Description
FINVID	Integer		Inventory ID
FINVNAME	Char	24	Inventory ID name
FVERSIONGKBID	Char	15	Version of the Global KB used to match
FIQDATE	Timestamp		Current® date that Inquisitor was run on Mainframe
FIQDATEFIRST	Timestamp		Date the Inquisitor was first run on Mainframe
FMIGDATEFIRST	Timestamp		Initial migrate date
FMIGDATE	Timestamp		Date Inquisitor data was loaded to Repository
FSYSCPUID	Char	12	CPU Serial number
FSYSPLEXID	Char	8	Sysplex Name
FSYSFMID	Char	8	System FMID
FSMFID	Char	20	SMF ID
FPREVINVID	Integer		Predecessor Inventory ID
FIQSYSPLEXUSE	Smallint		Was the SYSPLEX option turned on?
FVENDORCNT	Integer		Number of Vendors in Inventory
FPRODUCTCNT	Integer		Total Product count in Inventory ID
FLIBCNT	Integer		Total number of libraries in the inventory
FINVCONT	Char	1	Continuous Inventory
FINVTYPE	Smallint		Type of Inventory
FIQNAME	Varchar	254	IQ Filename and path for DB2 Schema name

Table 18. TINVREG

Column Name	Column Type	Column Length	Description
FREGIONID	Integer		Parent Region ID
FINVID	Integer		Inventory ID

Table 19. TIQHISTORY

Column Name	Column Type	Column Length	Description
FHISTORYID	Integer		Unique ID
FINVID	Integer		Inventory ID
FIQDATE	Timestamp		Inquisitor date
FMIGTYPE	Smallint		Migrate type

Table 19. TIQHISTORY (continued)

FIPADDR	Varchar	254	List of IP Addresses
FSUBNETMASK	Varchar	254	List of Subnet Masks
FMACADDR	Varchar	254	List of MAC Addresses
FOSNAME	Varchar	254	Operating System Name
FOSTYPE	Varchar	254	OS type e.g. ″WINNT″
FOSVERSION	Varchar	254	Operating System Version
FOSBUILDNUMBER	Varchar	254	Build Number of OS
FOSCSDVERSION	Varchar	254	Latest Service Pack Installed
FOSSERNUM	Varchar	254	OS product serial identification number.
FOSINSTDATE	Timestamp		Date of OS installation. Null if unknown.
FOSLASTBOOTDATE	Timestamp		Date of OS last boot. Null if unknown.
FOSLASTBOOTDATE	Timestamp		Date of OS last boot. Null if unknown.
FTOTALMEMORY	Integer		Total Physical Memory in MB

Table 20. TJOBDATA

Column Name	Column Type	Column Length	Description
FJOBNAME	Char	8	Job Name
FJOBID	Integer		Job ID
FJOBTYPE	Char	6	Job Type

Table 21. TLIBRARY

Column Name	Column Type	Column Length	Description
FLIBID	Integer		Library ID
FLIBNAME	Char	128	Library name
FINVID	Integer		Inventory ID
FCREATIONDATE	Timestamp		Library creation date on Mainframe
FLIBDEVNUM	Char	4	DASD device number
FREFERENCEDATE	Timestamp		Date library last referenced
FLIBVOLSER	Char	6	Volser library resides on
FTRACKSALLOC	Char	10	Number of allocated tracks
FTRACKSUSED	Char	10	Number of used tracks
FORIGIN	Char	1	Blank - PDS, E - PDSE, V - VTOC
FCATALOG	Char	1	S - SMS managed, C - Cataloged, U uncataloged W - cataloged on wrong volume
FLINKLIST	Char	1	Is this a link listed library?
FLINKPACK	Char	1	Is this library in the Linkpack
FAPFAUTH	Char	1	Is this library APF authorized

Table 21. TLIBRARY (continued)

FLASTUSAGE	Date		1st month of the most recent usage applied to any module in this library
FUSEFLAG	Smallint		Flag for library usage
FMODCNT	Integer		Number of modules in library
FOBSERVEFIRST	Timestamp		Date and time that library was first observed
FOBSERVELAST	Timestamp		Date and time that library was last observed
FOBSERVEDELETED	Timestamp		Date and time that library was deleted from Inquisitor data.

Table 22. TLPAR

Column Name	Column Type	Column Length	Description
FLPARID	Integer		LPAR ID
FLPARNAME	Char	20	Name of the LPAR
FUSEFLAG	Smallint		Indicates if usage has been attributed to this LPAR
FEDITFLAG	Smallint		Has this LPAR record been updated manually
FMANF	Char	10	Machine manufacturer
FMACHINE	Char	12	CPU Model
FSERIALNO	Char	12	CPU Serial number
FMIPS	Integer		Number of MIPS for LPAR
FSYSPLEXID	Char	8	Sysplex name if in a Sysplex

Table 23. TMODULE

Column Name	Column Type	Column Length	Description
FMODID	Integer		Module ID
FMODNAME	Char	40	Module name
FLIBID	Integer		Library ID
FPOVLIBID	Integer		Product Library ID
FMODFLAG	Smallint		Module indication flag as to which product version it belongs to and whether it has been superseded.
FFMID	Char	12	FMID
FMODSIZE	Char	8	Module size
FUSEFLAG	Smallint		Flag for module usage
FMODTYPE	Smallint		Type of module
FOBSERVEFIRST	Timestamp		Date and time that module was first observed
FOBSERVELAST	Timestamp		Date and time that module was last observed

Table 23. TMODULE (continued)

FOBSERVEDELETED	Timestamp		Date and time that module was deleted from Inquisitor data

Table 24. TPARAM

Column Name	Column Type	Column Length	Description
FKEY	Char	64	Parameter Key
FVALUE	Char	254	Parameter Value

Table 25. TPERIODS

Column Name	Column Type	Column Length	Description
FPERIOD	Date		Calendar month for usage
FINVID	Integer		Inventory ID
FSUMMARISED	Smallint		Summary status

Table 26. TPOVINV

Column Name	Column Type	Column Length	Description
FPOVINVID	Integer		Unique ID
FPOVID	Integer		Product ID
FINVID	Integer		Inventory ID
FPOVGID	Integer		Global Knowledge Base Version ID
FOBSERVEFIRST	Timestamp		First time observation was made
FOBSERVELAST	Timestamp		Last time observation was made
FOBSERVEDELETED	Timestamp		First time observation was not found in this library
FPRODUCTID	Integer		Product ID
FVENDORID	Integer		Vendor ID
FPRODINVID	Integer		Product and Inventory ID
FPATCHLIST	Varchar	254	List of current patches applied to z/OS UNIX product.

Table 27. TPOVLIB

Column Name	Column Type	Column Length	Description
FPOVLIBID	Integer		Unique ID
FPOVINVID	Integer		Product inventory ID
FLIBID	Integer		Library ID
FPOVLIBPID	Integer		Previous Product Version ID
FMATCHCODE	Char	3	Matching code
FMATCHID	Integer		Link to Inquisitor Decision table
FPRODUCTPCT	Integer		Product percentage used during match

Table 27. TPOVLIB (continued)

FVERSIONPCT	Integer		Version percentage used during match
FOBSERVEFIRST	Timestamp		First time observation was made
FOBSERVELAST	Timestamp		Last time observation was made
FOBSERVEDELETED	Timestamp		First time observation was not found in this library
FMODCNT	Integer		Number of load modules in library for product

Table 28. TPRODUCT

Column Name	Column Type	Column Length	Description
FPRODUCTID	Integer		Product ID
FPRODUCTNAME	Char	50	Product Name (could be alias name)
FGLOBALNAME	Char	50	Product Name (always Global Name if Alias is used)
FOPTIONNAME	Char	30	Option Name
FVENDORID	Integer		Vendor ID
FPRODSTATUS	Smallint		Billable Status
FCATEGORY	Char	30	Product Category
FDESCRIPTION	Varchar	254	Product Description

Table 29. TREGCLASS

Column Name	Column Type	Column Length	Description
FREGCLASSID	Smallint		Region Class ID
FCLASSNAME	Char	24	Region classification title
FICONID	Smallint		Icon Lookup

Table 30. TREGION

Column Name	Column Type	Column Length	Description
FREGIONID	Integer		Region ID
FREGIONNAME	Char	64	Region Name
FPARENTID	Integer		Owning Region ID
FREGCLASSID	Smallint		Region category
FSEQ	Smallint		Sequence number for regions

Table 31. TREGLEAF

Column Name	Column Type	Column Length	Description
FPARENTID	Integer		Parent Region ID
FREGIONID	Integer		Region ID

Table 32. TUIMPORTCTRL

Column Name	Column Type	Column Length	Description
FINVID1	Integer		Primary Inventory ID
FINVID2	Integer		Second ID
FINVID3	Integer		Third ID
FINVID4	Integer		Fourth ID
FINVID5	Integer		Fifth ID
FINVID6	Integer		Sixth ID
FINVID7	Integer		Seventh ID
FINVID8	Integer		Eighth ID
FMODVPOV	Char	1	If non 0 allows relaxed VPOV assignment
FLPARNAME	Char	20	LPAR name

Table 33. TUSELIB

Column Name	Column Type	Column Length	Description
FUSELIBID	Integer		Unique ID
FLPARID	Integer		LPAR ID
FLIBID	Integer		Library ID

Table 34. TUSEMTD

Column Name	Column Type	Column Length	Description
FMTDID	Integer		Unique ID
FLPARID	Integer		LPAR ID
FMODID	Integer		Module ID
FJOBID	Integer		Job ID
FUSERID	Integer		User ID
FPOVLIBID	Integer		Product Library ID
FEVENTCNT	Float		Total calls to module for this month
FPERIOD	Date		Calendar month that usage occurred
FFIRSTDATE	Date		First day of usage in the month
FLASTDATE	Date		Last day of usage in the month
FPROVIDER	Char	4	Provider Service
FPOVINVID	Integer		Unique ID
FPRODINVID	Integer		Product and Inventory ID
FACCOUNTID	Integer		Account ID

Table 35. TUSEPO

Column Name	Column Type	Column Length	Description
FUSEPOVINVID	Integer		Unique ID

Table 35. TUSEPO (continued)

FLARPID	Integer		LPAR ID
FPRODINVID	Integer		Product & inventory ID
FJOBCNT	Integer		Number of distinct Jobs for a product
FUSERCNT	Integer		Number of distinct Users for a product
FEVENTCNT1	Float		Sum of calls to this product current month
FEVENTCNT3	Float		Sum of calls to this product previous 3 month
FEVENTCNT6	Float		Sum of calls to this product previous 4-6 month
FEVENTCNT9	Float		Sum of calls to this product previous 7-9 month
FEVENTCNT12	Float		Sum of calls to this product previous 10-12 month
FPERIOD	Date		Calendar month in which usage occurred
FFIRSTUSED	Date		The earliest usage date in month
FLASTUSED	Date		The most recent usage date in month

Table 36. TUSEPOV

Column Name	Column Type	Column Length	Description
FUSEPOVINVID	Integer		Unique ID
FLARPID	Integer		LPAR ID
FPOVINVID	Integer		POVINV ID
FJOBCNT	Integer		Number of distinct Jobs for a product
FUSERCNT	Integer		Number of distinct Users for a product
FEVENTCNT1	Float		Sum of calls to this product current month
FEVENTCNT3	Float		Sum of calls to this product previous 3 month
FEVENTCNT6	Float		Sum of calls to this product previous 4-6 month
FEVENTCNT9	Float		Sum of calls to this product previous 7-9 month
FEVENTCNT12	Float		Sum of calls to this product previous 10-12 month
FPERIOD	Date		Calendar month in which usage occurred
FFIRSTUSED	Date		The earliest usage date in month
FLASTUSED	Date		The most recent usage date in month
FPRODINVID	Integer		Product Inventory ID

Table 36. TUSEPOV (continued)

FSEQNO	Smallint		Internal use only
FACCCNT	Integer		Number of distinct account codes

Table 37. TUSEPOVLIB

Column Name	Column Type	Column Length	Description
FUSEPOVLIBID	Integer		Unique ID
FLARPID	Integer		LPAR ID
FPOVLIBID	Integer		POVLIB ID
FJOBCNT	Integer		Number of distinct Jobs for a product
FUSERCNT	Integer		Number of distinct Users for a product
FEVENTCNT1	Float		Sum of calls to this product current month
FEVENTCNT3	Float		Sum of calls to this product previous 3 month
FEVENTCNT6	Float		Sum of calls to this product previous 4-6 month
FEVENTCNT9	Float		Sum of calls to this product previous 7-9 month
FEVENTCNT12	Float		Sum of calls to this product previous 10-12 month
FPERIOD	Date		Calendar month in which usage occurred
FFIRSTUSED	Date		The earliest usage date in month
FLASTUSED	Date		The most recent usage date in month
FSEQNO	Smallint		Internal use only
FACCCNT	Integer		Number of distinct account codes

Table 38. TUSERDATA

Column Name	Column Type	Column Length	Description
FUSERID	Integer		User ID
FUSERNAME	Char	10	User Name
FORGNAME	Char	8	Owning Organization
FREALNAME	Char	20	Real person′s name

Table 39. TVENDOR

Column Name	Column Type	Column Length	Description
FVENDORID	Integer		Vendor ID
FVENDORNAME	Char	50	Vendor Name (could be alias name)
FGLOBALNAME	Char	30	Vendor Name (always Global Name if Alias is used)

Table 39. TVENDOR (continued)

FVENDORGUID	Char	32	Vendor globally unique ID

Table 40. TVERSION

Column Name	Column Type	Column Length	Description
FPOVID	Integer		Version ID
FVERSIONNAME	Char	16	Version name
FPPNUMNAME	Char	16	PPNUM
FPRODUCTID	Integer		Product Option ID
FMINUSAGE	Float		Minimum usage threshold
FVERSIONGUID	Char	32	Product version globally unique identifier. PRODUCT.SW_KEY for SW_TYPE = ’VERSION’
FFEATUREGUID	Char	32	Product feature globally unique identifier. PRODUCT.SW_KEY for SW_TYPE = ’FEATURE’

Table 41. TACCOUNT

Column Name	Column Type	Column Length	Description
FACCOUNTID	Integer		Account ID
FACCOUNTCODE	Char	20	Job Account Code, truncated to 20 characters

Table 42. TUSEPRS

Column Name	**Column Type**	**Column Length**	**Description**
FUSEPRSID	Integer		Unique ID for TUSEPRS table
FREGVEND	Char	16	Product Registration Vendor name
FREGPROD	Char	16	Product Registration Product name
FREGFEAT	Char	16	Product Registration Feature name
FREGVRN	Char	6	Product Registration Version
FREGPID	Char	8	Product Registration Product identifier
FREGFLAGS	Char	8	Product Registration flags
FLPARID	Integer		LPAR ID
FPERIOD	Date		Calendar month when usage occurred
FFIRSTDATE	Date		The earliest usage date in month
FLASTDATE	Date		The most recent usage date in month

Table 43. NODE

Column Name	Column Type	Column Length	Description
NODE_KEY	Char	32	Global Unique ID (GUID) for this entry
NODE_TYPE	Char	4	Entry Type: HW or LPAR
HW_TYPE	Char	4	System z Hardware Type, for example 2096
HW_MODEL	Char	3	System z Hardware Model, for example P03
HW_PLANT	Char	2	System z Hardware Plant, for example 02
HW_SERIAL	Char	12	System z Hardware Serial, for example. 000000013EED
HW_NAME	Char	10	Configured Hardware Name
HW_VENDOR	Char	10	System z Hardware Vendor, for example IBM
LPAR_NUMBER	Char	4	Logical Partition Number, for example 1
LPAR_NAME	Char	10	Logical Partition Name, for example LPARSYS1
VMGUEST_NAME	Char	10	z/VM® Guest Name (if z/OS is running under z/VM)
HW_NODE_KEY	Char	32	NODE_KEY for related hardware parent
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated

Table 44. NODE_CAPACITY

Column Name	Column Type	Column Length	Description
NODE_KEY	Char	32	NODE GUID
PERIOD	Date		Month for this entry
START_TIME	Timestamp		First date that this entry is applicable for this Month
END_TIME	Timestamp		Last date that this entry is applicable for this Month
METRIC_TYPE	Char	10	Metric Type: MSU, SUBCAPMSTY
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated
QUANTITY	Integer		Metric Value

Table 45. PRODUCT

Column Name	Column Type	Column Length	Description
SW_KEY	Char	32	″Global Unique ID (GUID) for this entry. For SW_TYPE=VERSION this wll be the same value as VERSION_GUID For SW_TYPE=FEATYRE this wll be the same value as FEATURE_GUID″
SW_TYPE	Char	8	Entry type - VERSION or FEATURE
VENDOR_NAME	Char	50	Vendor name
PRODUCT_NAME	Char	50	Product name, which is a normalized form of Version Name in order to group different versions of products under the same product name
VERSION	Integer		Version
VERSION_NAME	Char	50	Product Version Title
FEATURE_NAME	Char	50	Product Feature Title
PID	Char	8	Product Identifier
EID	Char	8	Entitlement Identifier for the Product Feature
SSPID	Char	8	Subscription & Support Product Identifier
SSEID	Char	8	Subscription & Support Entitlement Identifier for the Product Feature
PRICETYPE	Char	10	Price Type (not used in 7.2)
SUBCAPACITY	Char	20	IPLA Subcapacity type: Execution-based, Reference-based, z/OS-based, Not eligible, NULL
ICA	Char	1	Y or N: IBM Company Agreement license
IPLA	Char	1	Y or N: International Program License Agreement
VUE	Char	8	IPLA Value Unit Exhibit
VENDOR_GUID	Char	32	Globally Unique ID for VENDOR_NAME
PRODUCT_GUID	Char	32	Globally Unique ID for VENDOR_NAME + PRODUCT_NAME
VERSION_GUID	Char	32	Globally Unique ID for VENDOR_NAME + VERSION_NAME + VERSION
FEATURE_GUID	Char	32	Globally Unique ID for VENDOR_NAME + VERSION_NAME + VERSION + FEATURE_NAME
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated

Table 46. PRODUCT_INSTALL

Column Name	Column Type	Column Length	Description
SW_KEY	Char	32	Product GUID
SYSTEM_KEY	Char	32	System GUID
INSTALL_DATE	Date		Date the product was first observed to be installed on this System
UNINSTALL_DATE	Date		Date the product was first observed to be missing from this System
LAST_USED_DATE	Date		Date the product was last used on this System
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated

Table 47. PRODUCT_NODE_CAPACITY

Column Name	Column Type	Column Length	Description
SW_KEY	Char	32	Product GUID
NODE_KEY	Date		Node GUID
PERIOD	Timestamp		Month for this entry
START_TIME	Timestamp		First date that this entry is applicable for this Month
END_TIME	Timestamp		Last date that this entry is applicable for this Month
METRIC_TYPE	Integer		Metric Type: INSTALLED, JOBNAMES, MODULES, USERS, SUBCAPMSU
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated
QUANTITY	Float		Metric Value

Table 48. SYSTEM

Column Name	Column Type	Column Length	Description
SYSTEM_KEY	Char	32	Global Unique ID (GUID) for this entry
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated
SID	Char	4	Product system ID. By default this is the SMFID. In cases where the same SMFID is used on different systems, the SID must be defined to a unique value for the customer enterprise in the Usage Monitor
SMFID	Char	4	z/OS SMF ID
SYSPLEX	Char	8	z/OS Sysplex name

Table 49. SYSTEM_NODE

Column Name	Column Type	Column Length	Description
SYSTEM_KEY	Char	32	System GUID

Table 49. SYSTEM_NODE (continued)

Column Name	Column Type	Column Length	Description
NODE_KEY	Char	32	Node GUID
PERIOD	Date		Month this entry is for
START_TIME	Timestamp		Time it was first observed that this system is using this Node in this month period
END_TIME	Timestamp		Time it was last observed that this system is using this Node in this month period
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated

Table 50. PRODUCT_USE

Column Name	Column Type	Column Length	Description
PERIOD	Date		Month for this entry
SYSTEM_KEY	Char	32	System GUID
SW_KEY	Char	32	Product GUID
FLPARID	Integer		TLPAR.FLPARID for convenient linking with PRODUCT_USE_DETAIL
HW_NODE_KEY	Char	32	NODE GUID for Hardware NODE that this System was last running on in this month
USER_CNT	Integer		MAX distinct Userid count
JOBNAME_CNT	Integer		MAX distinct Job Name count
ACCOUNT_CNT	Integer		MAX distinct Account Code count
SCRT_MSU			Sub-capacity Reporting Tool MSU (millions of service units per hour)
EVENT_CNT	Float		SUM of Module usage
START_DATE	Date		Date within this Period that usage was for first recorded
END_DATE	Date		Date within this Period that usage was for last recorded
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated

Table 51. PRODUCT_USE_DETAIL

Column Name	Column Type	Column Length	Description
PERIOD	Date		Month for this entry
FLPARID	Integer		System TLPAR.FLPARID for convenient linking with PRODUCT_USE
VERSION_GUID	Char	32	Product Version GUID
FEATURE_GUID	Char	32	Product Feature GUID
USERNAME	Char	8	User ID

Table 51. PRODUCT_USE_DETAIL (continued)

Column Name	**Column Type**	**Column Length**	**Description**
JOBNAME	Char	8	Job Name
ACCOUNTCODE	Char	20	First 20 chars of the Job Account Code
EVENT_CNT	Integer		SUM of Module usage
START_DATE	Date		Date within this Period that usage was for first recorded
END_DATE	Date		Date within this Period that usage was for last recorded

Appendix C. Deletion of obsolete libraries in Dorana Version 5 Release 4 Repository

The Repository tables might contain libraries prefixed with ″!″ (exclamation mark). As part of housekeeping, obsolete data associated with the libraries prefixed with ″!″ should be deleted.

The SQL statements listed here are to be used for the process of deletion. These SQL statements must be run in the sequence as shown and it is recommended that they be run as a batch job. For a sample batch job, make a copy of HSISTPRM from JCLLIB, and then replace the contents in //SQL.SYSIN DD with the SQL statements. For each SQL statement, replace REPschema with the Repository schema name at your site. Run time for this job depends on the number of records to be deleted. It is recommended to run this job during off-peak periods.

DELETE FROM REPschema.TUSEMTD WHERE FMODID in (Select TMODULE.FMODID FROM (REPschema.TLIBRARY AS TLIBRARY INNER JOIN REPschema.TMODULE as TMODULE ON TLIBRARY.FLIBID = TMODULE.FLIBID) WHERE TLIBRARY.FLIBNAME LIKE ′!%′) ;

DELETE FROM REPschema.TUSELIB WHERE FLIBID in (Select TLIBRARY.FLIBID FROM REPschema.TLIBRARY AS TLIBRARY WHERE TLIBRARY.FLIBNAME LIKE ′!%′);

DELETE FROM REPschema.TJOBDATA where FJOBID NOT IN (Select DISTINCT FJOBID from REPschema.TUSEMTD);

DELETE FROM REPschema.TUSERDATA where FUSERID NOT IN (Select DISTINCT FUSERID from REPschema.TUSEMTD);

DELETE FROM REPschema.TLPAR where FLPARID NOT IN (Select DISTINCT FLPARID from REPschema.TUSEMTD);

DELETE FROM REPschema.TPOVLIB WHERE FLIBID in (Select TLIBRARY.FLIBID FROM REPschema.TLIBRARY AS TLIBRARY WHERE TLIBRARY.FLIBNAME LIKE ′!%′);

DELETE FROM REPschema.TMODULE WHERE FLIBID in (Select TLIBRARY.FLIBID FROM REPschema.TLIBRARY AS TLIBRARY WHERE TLIBRARY.FLIBNAME LIKE ′!%′);

DELETE FROM REPschema.TLIBRARY WHERE REPschema.TLIBRARY.FLIBNAME LIKE ′!%′

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe 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 grant 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 character set (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.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

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 might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes are incorporated in new editions of the publication. IBM may make improvements and 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.

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 in softcopy form, the photographs and color illustrations might not display.

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 www.ibm.com/legal/copytrade.shtml.

Intel is a registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Related publications

Publications referenced in this guide are listed here:

- *OS/390 V2R10.0 MVS Programming: Assembler Services Guide (GC28-1762-09)*
- *Language Environment Programming Reference (SA22–7562).*
- *Language Environment Programming Guide (SA22–7561)*
- *Language Environment Customization (SA22-7564).*

Supplied on DVD

- *IBM Tivoli Common Reporting (LCD7 - 1995–02)*

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully.

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

- Use assistive technologies, such as screen-reader software and digital speech synthesizer, to hear what is displayed on the screen. Consult the product documentation of the assistive technology for details on using those technologies with this product.
- Operate specific or equivalent features using only the keyboard.
- Magnify what is displayed on the screen.

Magnifying what is displayed on the screen

You can enlarge information on the product windows using facilities provided by the operating systems on which the product is run. For example, in a Microsoft Windows environment, you can lower the resolution of the screen to enlarge the font sizes of the text on the screen. Refer to the documentation provided by your operating system for more information.

Index

Part Number: 5698–B39

Printed in USA

SC27-2701-00