IBM Z Software Asset Management 9.1 Administration Guide and Reference

SC22-5474-05

Contents

Product overview **1**
- **Implementing IBM Z Software Asset Management with SQLite database** 3
- **Accessibility** 4
- **How to read the syntax diagrams** 4
- **What's new in 9.1** 7

Planning for deployment **9**
- **Predeployment considerations** 9
- **Deployment data processes** 10
- **Deployment for a single Repository** 10
- **Deployment for multiple Repositories** 11

Installing and customizing IBM Z Software Asset Management **12**
- **Installation prerequisites** 12
- **System configuration requirements** 13
- **Security and authorization requirements** 13
- **Checklist of installation and customization tasks** 15
  - **Installing target libraries** 18
  - **Preparing Db2 database prerequisites** 19
  - **Preparing local environment settings** 19
  - **Configuring a test Repository database** 23
  - **Configuring a production Repository database** 25

Implementing deployment scenarios **28**
- **Scenario 1: Implementing a single Repository database with a single GKB database** 28
- **Scenario 2: Implementing multiple Repositories with a shared GKB database** 29
- **Scenario 3: Implementing multiple Repositories with multiple GKB databases** 31
- **Scenario 4: Collecting and transferring Inquisitor and usage data from remote sites** 33
- **Scenario 5: Implementing in a sysplex environment** 33

Migrating to IBM Z Software Asset Management 9.1 ..... **34**
- **Migrating to IBM Z Software Asset Management from an earlier version** 34
  - **Migrating from IBM Z Software Asset Management 8.3/8.3.1 to IBM Z Software Asset Management 9.1 (Db2® database)** 35
  - **Migrating from IBM Z Software Asset Management 8.3/8.3.1 to IBM Z Software Asset Management 9.1 (SQLite database)** 38
  - **Migrating from IBM Z Software Asset Management 8.2 to IBM Z Software Asset Management 9.1 (Db2 database)** 41
  - **Migrating from IBM Z Software Asset Management 8.2 to IBM Z Software Asset Management 9.1 (SQLite database)** 45
  - **Post migration errors** 47

Collecting and importing data with IBM Z Software Asset Management **48**
- **Updating the Global Knowledge Base** 49
- **Applying Updates to GKB Database Via Fix Central** 49
- **Collecting scanned libraries with the Inquisitor for z/OS** 50
  - **Running the Inquisitor program** 50
  - **PLX parameter of the Inquisitor program** 50
  - **Collecting SMP/E maintenance information** 51
  - **Inquisitor program parameters and files** 52

Inquisitor program command syntax..... 54
Inquisitor examples.....61
Designing Inquisitor requests..... 63
Controlling the data extraction level..... 64
Extracting LE compile unit information..... 64
Scanning migrated libraries.....64
Scanning generation data sets..... 65
Logical Volume Mapping.....65
Collecting information about the I/O configuration..... 69
Inquisitor completion codes..... 70
Collecting UNIX files with the Inquisitor for z/OS UNIX.....70
Inquisitor for z/OS UNIX overview..... 70
Running the Inquisitor for z/OS UNIX program.....71
PLX parameter of the z/OS® UNIX® Inquisitor program.....71
Inquisitor for z/OS UNIX program parameters and files..... 72
z/OS UNIX Inquisitor example..... 73
Security considerations.....74
Collecting usage data with the Usage Monitor..... 74
Setting up the Usage Monitor..... 74
Starting and stopping the Usage Monitor.....79
Refresh processing for the Usage Monitor.....80
Usage Monitor general commands..... 80
Usage Monitor Trace Facility.....101
IBM CICS® Transaction Server for z/OS®..... 108
Customizing a CICS region to provide usage data..... 109
Importing Inquisitor data.....110
Running the Inquisitor Import..... 110
Import filters and matching..... 111
TPARAM parameters..... 111
Importing Usage data..... 112
Running the Usage Import.....113
TPARAM parameters..... 113
Aggregating usage and discovery data..... 114
TPARAM parameters..... 114
Activating the Automation Server..... 115
Automation Server overview.....115
Running the Automation Server.....116
Automation Server control data set maintenance..... 122
Automation Server Utility program.....122

Reporting with the Analyzer.....127
Analyzer prerequisites.....128
Analyzer JCLLIB and PARMLIB members.....128
Running the Analyzer in online mode..... 129
Analyzer communication port..... 130
Analyzer inactivity timeout.....130
Analyzer security..... 131
Analyzer BASIC security.....131
Analyzer SYSTEM and SYSTEM-TLS security..... 132
SSL Certificates.....134
Online login to the Analyzer.....136
Controlling the Analyzer address space.....140
Running the Analyzer in batch mode..... 140

REST API.....141
Prerequisites..... 141
Installing the REST API..... 142

Expected Result ... 143
Authorization ... 144
Errors and Workarounds ... 144
HTTPS ... 144
Graceful Server Shutdown Via SDSF ... 145
End Points ... 145

Running utilities ... 149
Condensing usage data with the ZCAT utility ... 149
Deleting usage data with the Usage Deletion utility ... 152
Deleting a specific system with the System Deletion utility ... 153
Deleting obsolete data with the Physical Deletion utility ... 153
Deleting obsolete audit records with the MDEL Deletion utility ... 154
High-level qualifier listing ... 155
Updating the TPARAM table ... 156
Tagging unidentified products with the Product Tagging utility ... 156
Product tagging process ... 156
Product tagging job and control statements ... 157
Product tagging examples ... 158
Importing Subcapacity reporting data with the SCRT Import utility ... 159
Extracting data with the XML Export utility ... 161
Transferring output XML by FTP ... 161
Loading data into a staging table (for Db2 only) ... 162
Reporting LMOD ownership from the CSI ... 162
Compressing and decompressing data sets with the HSIZIP utility ... 163
Text data processing ... 163
Binary data processing ... 163
HSIZIP program parameters ... 164
HSIZIP files ... 167
Dynamic invocation by other programs ... 167
HSIZIP data set support ... 167
HSIZIP return codes ... 168
Browsing zipped data ... 168
Browsing active collection data ... 169
Verifying database changes since the product was released ... 170

Configuring language support ... 170
Configuring Japanese messages ... 170
Enabling the Analyzer utility for Japanese ... 171
Configuring the Japanese Db2 subsystem for use with IBM Z Software Asset Management ... 171
Configuring the Turkish Db2 subsystem for use with IBM Z Software Asset Management ... 174

Reference information for IBM Z Software Asset Management ... 174
Repository table layouts ... 174
Post-installation jobs ... 200
Jobs generated in JCLLIB for a Db2 environment ... 200
Jobs generated in JCLLIB for a remote environment ... 204
Jobs generated in JCLLIB for an SQLite environment ... 204
Database performance and tuning ... 208
Db2 database ... 208
SQLite database ... 209
Database resources used by IBM Z Software Asset Management ... 209

Troubleshooting and support ... 213
Troubleshooting a problem ... 213
Problems and solutions ... 214

IBM Z Software Asset Management messages..... 218
HSIA - Automation Server messages.....218
HSIC - Operation messages..... 222
Error codes..... 231
HSII - REXX utility messages.....235
HSIP - Inquisitor for z/OS messages and codes..... 243
HSIT - Product tagging messages..... 265
HSIX - Inquisitor for z/OS UNIX messages and codes.....272
HSIZ - Usage Monitor messages..... 275

Appendix A: Description of output message content..... 299
Notices.....303
Trademarks.....305
Terms and conditions for product documentation.....305

Index..... 306

Overview of IBM Z Software Asset Management

IBM Z Software Asset Management is built on the concept of remote and central mainframe components which work together to produce reports on z/OS® mainframe products and their usage. This section provides you with a high-level overview of the IBM Z Software Asset Management core architecture.

IBM Z Software Asset Management runs on z/Architecture® mainframes that use the z/OS operating system. Its purpose is to:

- Discover and identify products for the z/OS platform.
- Monitor software usage and trends.
- Report on the MSU capacity of each system under which the product runs.
- Provide reporting functions 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 a z/OS environment, software is contained in load libraries or as z/OS UNIX executable files. IBM Z Software Asset Management scans contents of these libraries and executable files to determine which software products are installed. IBM Z Software Asset Management also monitors the loaded programs and executable files to measure software usage.

The discovered load libraries and executable files are then checked against a global database of product information. IBM Z Software Asset Management uses this information to determine which products are installed and used on each system.

The IBM Z Software Asset Management Usage Monitor gathers information about events for modules and executable files which are then attributed to each product.

The workflow is illustrated in Figure 1 on page 2, followed by a brief description of the components.

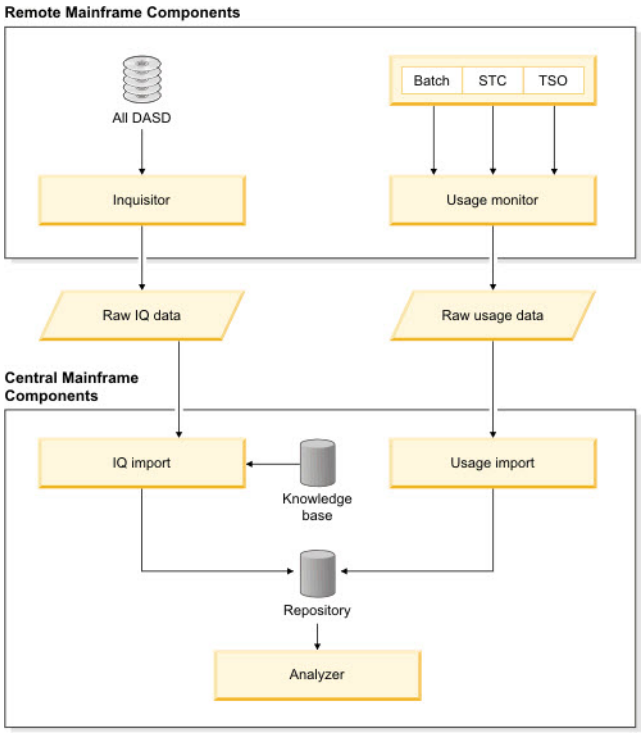


Figure 1. Product workflow

Inquisitor

The Inquisitor is a batch job that discovers loadable programs in z/OS data sets and z/OS UNIX System Services file systems. A program locates load libraries on z/OS DASD devices and captures information about the load modules. The process can be targeted to specific devices, libraries, or groups of libraries. The program creates a compressed data set, which is then used as input to the Inquisitor Import procedure.

An additional process locates and scans z/OS UNIX directories for program objects and captures this information. The process creates a compressed data set that is then used as input to the Inquisitor Import for z/OS UNIX procedure.

Usage Monitor

The Usage Monitor is a started task or batch job that monitors and records loaded modules of batch jobs, started tasks, TSO users, and z/OS UNIX executable files.

Knowledge Base

The Global Knowledge Base (GKB) is a database that is provided with IBM Z Software Asset Management. The GKB has a list of all z/OS globally-identified products that are used by the product in the process of matching.

Inquisitor (IQ) Import

The Inquisitor Import is a batch job that loads Inquisitor data into database tables on z/OS for z/OS load modules and z/OS UNIX program objects. The imported Inquisitor data is then matched against the Global Knowledge Base.

Usage Import

The Usage Import is a batch job that imports Usage Monitor data into the Repository. The data is matched against load modules and z/OS UNIX executable files and the data is then aggregated with installed software products. After this process has been completed, you can view the usage data with the IBM Z Software Asset Management Analyzer reports.

Repository

The Repository is a set of database tables for z/OS data that stores information about the software products discovered and their usage data.

Analyzer

The Analyzer queries the IBM Z Software Asset Management database and displays Analyzer online reports. The Analyzer runs as a started task or batch job on the same z/OS system where the Db2® subsystem or SQLite database runs. The output formats can be XML, HTML, Excel, text, or comma separated value (CSV). You can logon to the Analyzer from a web browser to display interactive reports. You can also run the Analyzer in batch mode and save the results to an output data set on z/OS.

Process flow

Data collected on the target systems by the Inquisitor and the Usage Monitor batch programs, are then imported into the Repository database tables. The database is located on a z/OS system within a Db2 subsystem or an SQLite database.

Following is a summary of the workflow tasks:

1. Importing and matching the data collected by the Inquisitor.
2. Importing the collected usage data into the Repository.
3. Running utilities to manage and maintain your data. This task is optional.
4. Reporting using the Analyzer, which consists of online and batch components.

Implementing IBM Z Software Asset Management with SQLite database

You can implement IBM Z Software Asset Management with an SQLite database if you do not have a Db2 for z/OS license, or if you are currently unable to deploy one. If you implement IBM Z Software Asset Management with an SQLite database, there are certain functional and performance limitations.

SQLite is an in-process library that implements a self-contained, serverless, transactional SQL database engine. It is an embedded SQL database engine that is unlike many other SQL databases, because it does not have a separate server process. SQLite reads and writes data directly to ordinary disk files. When you implement SQLite with IBM Z Software Asset Management , the complete SQL database is contained within a single z/OS UNIX zFS file.

Limitations

IBM Z Software Asset Management with SQLite database is implemented with the following configuration:

- 500 products identified from 15 LPARs
- 3 months usage data (about 6 million usage records)
- Only 1 repository within a zFS file

This configuration represents the limitation for SQLite support with IBM Z Software Asset Management . Only one repository within a zFS file is supported. If you require a database with a larger configuration supporting multiple repositories, consider implementing IBM Z Software Asset Management with Db2 for z/OS.

SQLite has limited concurrency because it uses read/write locks on the entire database file. Therefore, if any process is reading from any part of the database, all other processes are prevented from writing to any other part of the database. Similarly, if any one process is writing to the database, all other processes are prevented from reading any other part of the database.

Accessibility

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

System console

You can set up, run, and maintain IBM Z Software Asset Management using a 3270 emulator as the system console. IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities.

Reporting

The online version of the Analyzer reporting utility provided by IBM Z Software Asset Management supports the standard keyboard commands that are used by screen reader applications. You can generate reports in a variety of accessible formats, including HTML and Excel file formats.

Images

All images in the information center that convey additional information have alternate text.

Tables

You can navigate the content of all tables in the information center with a screen reader application.

Syntax diagrams

Syntax diagrams in the information center are displayed as railroad track images. Each syntax diagram has a link to a dotted decimal version of the diagram that can be read by a screen reader application.

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see the IBM Human Ability and Accessibility Center. The IBM Human Ability and Accessibility Center is at the following web address:

http://www.ibm.com/able (opens in a new browser window or tab)

How to read the syntax diagrams

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

Figure 2 on page 5 shows a sample syntax diagram that includes the various notations that are 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.

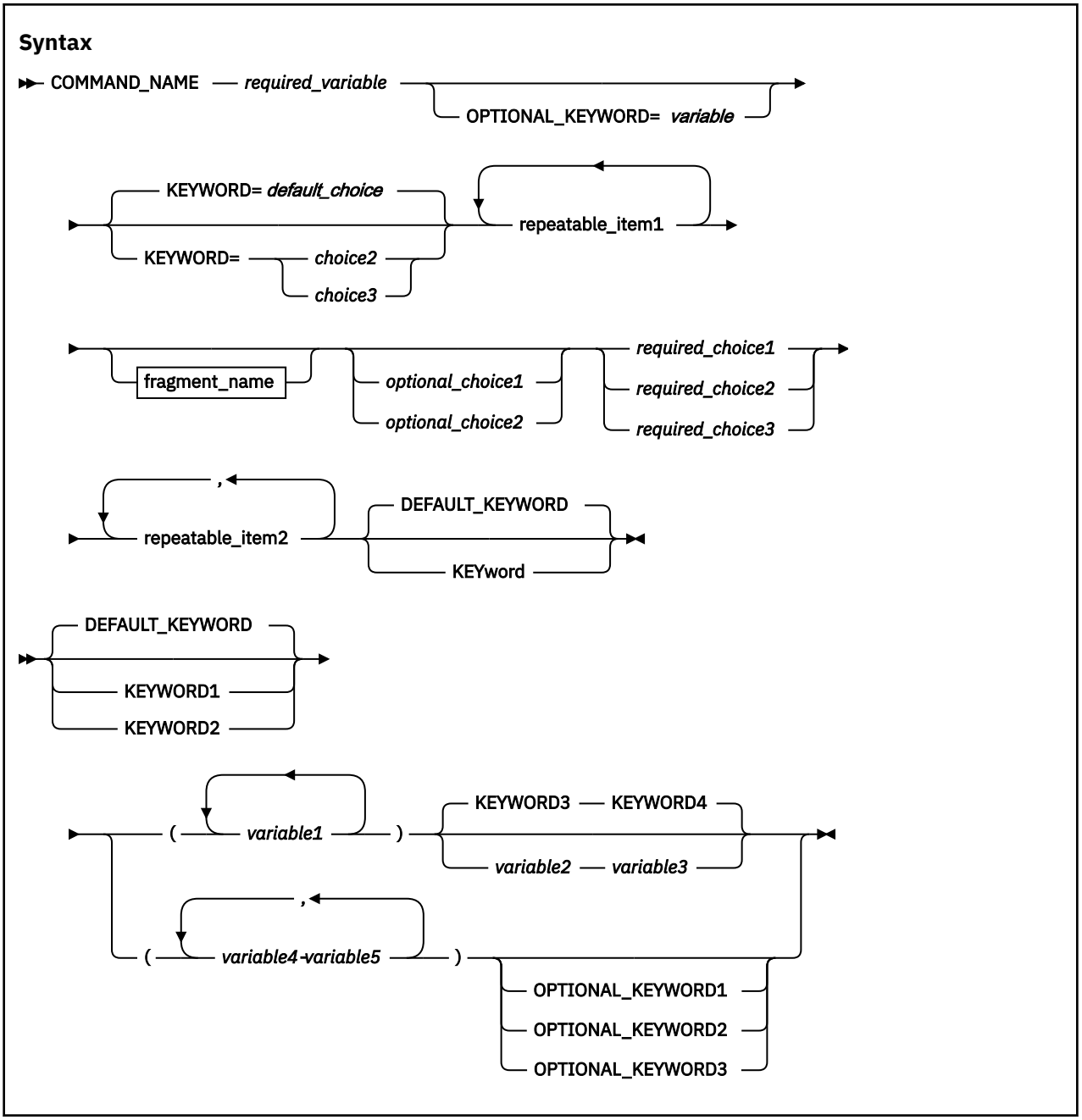


Figure 2. Sample syntax diagram

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.

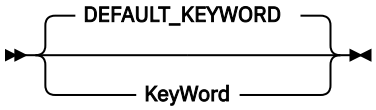
The ►—— symbol indicates that a statement is continued from the previous line.

The ——►◄ symbol indicates the end of a statement.

Keywords

Keywords appear in uppercase letters.

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

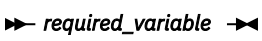


In this example, you could type "KW" or "KEYWORD".

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

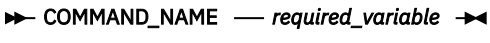
Variables

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



Required items

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



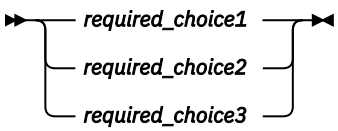
Optional items

Optional items appear below the main path.

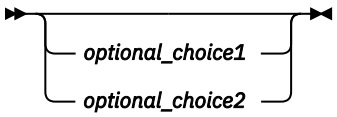
Choice of items

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

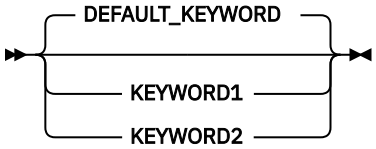
If you *must* choose one of the items, one item of the stack appears on the main path.



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.

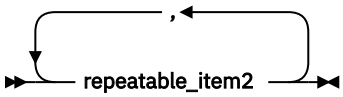


Repeatable

items

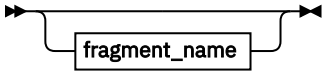
An arrow returning to the left above the main line indicates an item that can be repeated.

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.



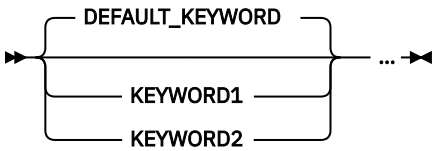
Fragments

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



⋮

fragment_name



What's new in 9.1

The new features and capabilities in IBM Z Software Asset Management 9.1 help your organization achieve greater efficiency in asset discovery.

API Support Enabled

An API Server has been added that enables data that resides in a IBM repository to be extracted as JSON files. The initial release has 5 endpoints:

- Product Inventory
- Product Use by Month
- End of Service report
- Product use by Machine
- Discovered Product Detail

UNIX License Information Added to UNIX GKB

IBM Z Software Asset Management 9.1 now has License information just like z/OS for UNIX products. This means, in the Asset reports, the UNIX products will now show zPricer details just like z/OS products.

Product Suites Added for UNIX

Support for UNIX product Suites has been included.

Ability to Exclude Product Suite Processing

A new parameter has been added to IQIMPORT that enables users to exclude the processing of product suites. The default is to process Product Suites.

Ability to Post Notifications in Analyzer

A new function has been added to Analyzer, allowing notifications to be posted. When a user first logs onto Analyzer, they will see a list of current unread notifications. Users can always view the report at any time. Additionally, users with administrator privileges can easily post new notifications from a dedicated report.

IQ and Usage Logs Displays IBM Version and APAR

Both IQ and Usage log reports will now show the ZSAM version the data came from and the APAR level of the code.

New Report to View Products by Categories

A new report has been added that will allow users to view products by category. This will show products based on category and show multiple products under the same category.

GKB Updates via FixCentral

Optionally, you can also download GKB updates via FixCentral as well as PTF.

Ability to View Data by Operating System for few Reports

Discovered Product Detail, Discovered Product by System, and Product Inventory reports now allow viewing of products by operating system.

EOS Date Added to Discovered Product Detail Report

The End of Service date has been included to the Discovered product Detail report.

LKB Summary Report Versions

The LKB Summary report now shows the versions of the LKB products that have been added.

GKB Summary Report Supports UNIX Asset Information

The GKB Summary report now supports UNIX Asset information. The report will now also display the Category the product belongs to.

Staging Table Added to SQILITE

In SQLITE, it’s now possible to pre-load usage data into a temporary table for improved viewing of raw use data by report Usage Monitor Detail report. Since the data is now indexed, search time has been improved.

Inquisitor JCL Changes

JCL DD names HSIPZIP, HSIPOUT, HSIPDMP, and HSIDAMAP are changed to INQPZIP, INQPOUT, INQPDMP, and DASDMAP, respectively.

JCL DD names HSIXZIP, HSIXOUT, HSIXMSG, HSIXROOT, and HSIXOMIT are changed to INQXZIP, INQXOUT, INQXMSG, INQXROOT, and INQXOMIT respectively.

Product Tagger JCL Changes

JCL DD name HSIREDIR is changed to TAGREDIR.

Usage Monitor Changes

JCL DD names HSIZMSG, HSIZIN, HSIZSNAP, HSIZDATA and HSIDAMAP are changed to UMONMSG, UMONIN, UMONSNAP, UMONDATA and DASDMAP respectively.The HSIIPEEK REXX EXEC shipped in the SHSIEXEC library has been updated to reflect the new DD name.

The CRO and RDC settings have been removed due to the lack of any observable performance benefits.

New Container Product Version

The Container Product version enables you to run the Mainframe collectors as usual, with the added benefit of executing report analysis, Usage, and IQ data imports on a zLinux or zCX system instead of z/OS. This update helps significantly reduce your CP MSU consumption for IBM Z Software Asset Management use.

The new Container package can be downloaded from FixCentral. Look for IBM Z Software Asset Management under the Tivoli Brand.

Planning for deployment

Before you deploy IBM Z Software Asset Management , consider which deployment option is best suited to your environment.

Related tasks

Implementing deployment scenarios

Most implementations of IBM Z Software Asset Management are based on one of the common deployment scenarios. An example is provided for implementing each of these common deployment scenarios with a Db2 Repository database. You can adapt an example for use with an SQLite Repository database.

Predeployment considerations

You can deploy IBM Z Software Asset Management to use a single Repository or multiple Repositories.

Implementing a deployment in a single Repository is straightforward because the data from all systems is imported into a single Repository. Before you deploy with a single Repository, plan the following aspects of the deployment:

- How frequently will the Inquisitor scan each system?
- If you have systems that are located at remote sites, what mechanism will transfer collected Inquisitor and usage data through file transfer protocol (FTP) to the central site, and how frequently will these transfers occur?
- How often is it necessary to load Inquisitor and usage data from each system into the Repository?

To deploy with multiple Repositories, plan the following aspects of the deployment:

- How frequently will the Inquisitor scan each system?
- How many Repositories to include in the deployment, and are all these Repositories located at the central site?
- For each system, which Repository loads the Inquisitor and usage data for the system?
- If you have systems that are located at remote sites, what mechanism will transfer collected Inquisitor and usage data through file transfer protocol (FTP) to the central site, and how frequently will these transfers occur?

- How often is it necessary to load Inquisitor and usage data from each system into its specified Repository?

Deployment data processes

IBM Z Software Asset Management is structured on several key data processes.

Inquisitor data
The Inquisitor scans DASD volumes for libraries containing load modules and HFS/zFS files for z/OS UNIX program objects and produces Inquisitor data. These load modules and program objects are matched and associated to a particular vendor and product and the matched information is then loaded into the Repository tables. These processes are performed by running the Inquisitor Import job.

Usage event
A usage event describes a unique load of a load module or program object for an address space that can contain an account code. The Usage Monitor records these usage events as they occur on a particular operating system image. After the usage data is imported into the Repository, each usage event is identified by the load module name, library name, and volume. It can then be associated to a particular product discovered on that system.

Repository
The Repository is a collection of database tables that contain processed Inquisitor and Usage Monitor data. To ensure that accurate data is stored in the Repository tables, the following criteria must be met:

- The DASD VOLSERs of the data being imported must be unique unless the DASD VOLSERs are shared or are clones of each other with identical contents.
- The data imported must be from systems with unique SMF IDs.

When you are designing the scope of a Repository, there are a few common scenarios that most installations fit into. It is common to define the scope of a Repository based upon a data center. In this scenario, each data center in the organization has a separate Repository.

CAUTION: Import only DASD volumes with a unique VOLSER into your Repository.

The only way to prevent this sharing from taking place is to divide the z/OS systems with conflicting DASD/SMF IDs into separate Repositories. This can entail running one Repository for each sysplex or stand-alone z/OS system. With IBM Z Software Asset Management , it is common for IT service providers to define separate Repositories for each customer. This definition also satisfies the need for separation of data and ease of reporting.

It is recommended to have a central Db2 subsystem or SQLite databases that contain all the Repositories in your entire enterprise. The usage and Inquisitor data that require processing should be transmitted to this central Db2 subsystem or SQLite database by using the IBM Z Software Asset Management Automation Server or equivalent automation product.

Deployment for a single Repository

The recommended procedure for deploying the Inquisitor and Usage Monitor to collect raw data is to deploy both components on every system in your organization.

After you deploy both components to each system in your organization, perform data collection in the following sequence:

1. Use the Inquisitor Job to scan all available DASD on each z/OS system.
2. Import Inquisitor data by running the Inquisitor Import job.
3. Ensure that the Usage Monitor is active on all z/OS systems, directly after IPL.
4. Import Usage data by running the Usage Import job. Run this job after Inquisitor data has been imported.

IBM Z Software Asset Management displays products that have been discovered. Usage data collected from every system by the Usage Monitor is imported and usage events are assigned to the discovered products, enabling analysis of product use by system.

The first step in deploying IBM Z Software Asset Management is to run the SMP/E (System Modification Program/Extended) installation of the product, followed by the customization and creation of the database resources.

The next step is to create a test Repository. This deployment exercise is useful as it helps you to:

- Gain familiarity with the product.
- Check that your Repositories are defined correctly in terms of your business requirements and that the DASD VOLSERs and SMF IDs are unique.
- Ensure that data-sizing is adequate.
- Analyze the integrity of the data.

As part of this test implementation, you can then deploy the Inquisitor and Usage Monitor to all systems in your organization. It is advisable to first start the Usage Monitor on every system, in order to gather a significant amount of usage data. Place the test repository on a test or development Db2 subsystem.

At this point you can start the IBM Z Software Asset Management Analyzer and connect to the Repository. To verify the data collected by the Inquisitor and Usage Monitor, log on to the Analyzer and navigate to the Discovery menu tab. From this menu you can proceed to various reports on discovered products and module usage.

After you move your Repositories to their final location, you should consider setting up automation of the product.

Deployment for multiple Repositories

Multiple Repositories can be required to provide support for more than one data center, for different geographical regions, and for running multiple customers.

Db2 only: It is recommended that not more than 50 repositories be defined in a Db2 subsystem that is referenced by each Analyzer. This is to prevent overload to the Analyzer and also for easier management of repositories.

You can locate multiple Repositories in one central location, or you can locate them in geographically dispersed locations. Multiple Repositories may be organized as follows:

1. For a central location
2. For geographically dispersed locations

Central location

Each Repository contains data that is divided up into logical units, for example:

- Data center
- Outsourced customer
- Sysplex

Each Repository has its own database. For Db2, all repositories must reside in the same Db2 subsystem. For SQLite, each repository must reside in its own SQLite database. For Db2 only, the advantage of this configuration is that reporting can be performed on data across all repositories. With this configuration, all repositories can share the same Global Knowledge Base (GKB) and you only have to maintain a single copy of the GKB.

Geographically dispersed locations

Each Repository is defined with its own database at a specific geographic site as a stand-alone operation. Reporting can only be performed for each specific Repository. The disadvantage with this configuration is that it can be necessary to consolidate Repository data to a central site for reporting purposes.

Installing and customizing IBM Z Software Asset Management

The product installation involves downloading the product and available updates, preparing the database, and configuring and populating a test database. After verifying that all components are correctly installed, you duplicate the test database to create a production database where you automate data collection and import tasks.

Installation prerequisites

Before you install IBM Z Software Asset Management, verify that the required hardware and software are available in the installation environment.

Hardware requirements

IBM Z Software Asset Management requires a z/Architecture machine capable of running z/OS 2.5, or later.

Software requirements

The software required for IBM Z Software Asset Management is:

- z/OS 2.5 or later with z/OS UNIX System Services configured.
- Database software: Either Db2 or SQLite. It is not necessary to install the database on all of your z/OS systems, but it must be installed on at least one z/OS system.

  If you choose Db2 for your IBM Z Software Asset Management database, either of the following:

  - Db2 for z/OS 12.1 and Db2 Utilities for z/OS
  - Db2 for z/OS 13.1 and Db2 Utilities for z/OS

  If you do not have a Db2 license:

  - SQLite 3.45.1, that is embedded in IBM Z Software Asset Management

    Contact IBM support in order to install IBM Z Software Asset Management with SQLite only.
- Language Environment® for z/OS
- Any of the following browsers:
  - Firefox Standard Release 128.0.3 with JavaScript and cookies enabled
  - Firefox Extended Support Release 128.0 with JavaScript and cookies enabled
  - Microsoft Edge 127.0.2651.86 with JavaScript and cookies enabled
  - Chrome 127.0.6533.100
- Microsoft Excel 2003

System configuration requirements

SMF

The Usage Monitor collects several types of information from SMF records written by the system. It does this by deploying its own IEFU84 SMF exit to gain access to type 30 and type 89 SMF records.

However, this exit will only be given control if IEFU84 is an active SMF exit.

Accordingly, in the system's SMF parameters, ensure that NOEXITS is not specified, and that IEFU84 is specified in any list of SMF exits named in any EXITS parameter which may apply to the started task (STC) or batch job (JES2 or JES3) subsystems.

Also ensure that type 30 and type 89 SMF records are being written by the system, and that SMF interval recording is active for the started task (STC) and batch job (JES2 or JES3) subsystems.

Inquisitor virtual storage requirements

If you use IEFUSI or SMFLIMxx in PARMLIB to limit data space storage usage, ensure that the Inquisitor can create up to 5 plus MAXTASKS data spaces, and that up to 12GB of data spaces may be allocated. These sizes are a small fraction of the z/OS default limits. The Inquisitor reports data space usage details to provide performance feedback.

Security and authorization requirements

A z/OS user ID with appropriate RACF® access is required to submit the batch jobs used in the customizing and operation of IBM Z Software Asset Management. Additional security and authorization configurations can be necessary, depending on your environment.

RACF authorizations

The following table lists the RACF authority required to run IBM Z Software Asset Management Started Tasks, Usage Monitor, Analyzer, and Automation Server. Consult with your RACF administrator to define the required RACF authority.

Table 1. RACF data set access required by each started task

Started task name	SHSIMOD1	PARMLIB	SHSIANL1	SHSIANL2	ACDS	(Db2 only) SDSNLOAD and SDSNEXIT	HLQIDS data set	Usage Monitor output data sets
Usage Monitor	READ	READ	n/a	n/a	n/a	n/a	READ	ALTER
Analyzer	READ	READ	READ	READ	n/a	READ	n/a	n/a
Automation Server	READ	READ	n/a	n/a	CONTROL	n/a	n/a	n/a

The started task should be defined in the resource class STARTED, with additional detail in the STDATA segment of the resource. It can also be defined in the started task table ICHRIN03, but this requires an IPL to add or update a task definition. For example:

```
RDEFINE STARTED HSI*.* UACC(NONE)  +
                  STDATA (USER(uuuuuuu))
```

Replace *uuuuuuu* with the name of the started task user for IBM Z Software Asset Management.

```
SETROPTS RACLIST(STARTED) REFRESH
```

When SECURITY=SYSTEM is set for the Analyzer, the application name of HSICANLZ is supplied to SAF during authentication. Security administrators can use permissions to the HSICANLZ resource in the APPL RACF class to control which users are allowed to logon to the Analyzer.

For non-RACF security products, consult your Security Administrator.

z/OS UNIX security

Both the Usage Monitor and the z/OS UNIX Inquisitor need sufficient authority to navigate the UNIX file system. The writer task of the Usage Monitor calls the realpath service for all collected path names to resolve symbolic links, while the UNIX Inquisitor is tasked with discovering executable files.

The HSIPHOST module is called by the Usage Monitor writer task and by both Inquisitor programs to collect the system TCP/IP host name and IP address. This action requires a security user profile which has an associated UNIX UID value. The call of HSIPHOST can be disabled by relevant Usage Monitor and Inquisitor settings, if necessary.

APF

The Inquisitor and Usage Monitor use z/OS authorized system services. These programs are contained in the PDSE Load Library SHSIMOD1, which must be authorized using APF in order to run the Usage Monitor and/or the Inquisitor when the latter is not being run with PARM=NOAPF.

The Analyzer also requires an APF authorized environment when the SECURITY=SYSTEM option is selected so that it can issue the relevant SAF calls required for user authentication.

MEMLIMIT

The Usage Monitor creates memory objects, which are areas of virtual storage that have addresses greater than 2GB and can only be addressed in 64-bit addressing mode.

The MEMLIMIT setting, which applies to the Usage Monitor address space, must be set at a value high enough to allow the Usage Monitor to create all memory objects necessary for operations. It is recommended that MEMLIMIT=NOLIMIT is used for the Usage Monitor address space.

The actual size of the memory objects that the Usage Monitor creates depends on the SIZ, QSZ, and ASZ settings.

Dynamic exits

Ensure that the EXITS clause in the SMFPRMxx PARMLIB settings that apply to started tasks (STC subsystem) and batch jobs (JES2 or JES3 subsystem) explicitly list IEFU84. The Usage Monitor will install its own IEFU84 SMF exit.

The Usage Monitor will also install its own BPX_POST_SYSCALL system exit. If you want to track usage of Java software fetched from JAR files, then list BPX1OPN (or BPX4OPN) in the EBCDIC text file named in the SC_EXITTABLE parameter of active BPXPRMxx PARMLIB settings.

Db2 authorization

You need Db2 privileges to perform the following tasks:

- DBADM authority to access the product database. You may need to drop and create Db2 resources.
- BIND plans and packages
- EXECUTE authority to execute plans and packages
- SELECT authority to access the Db2 Catalog tables
- LOAD, REPAIR, and STATS privileges to run Db2 utilities LOAD, REPAIR, and RUNSTATS
- GRANT USE OF BUFFERPOOL privilege to use specific buffer pools
- GRANT USE of STOGROUP privilege to use a specific storage group
- ALTER BUFFERPOOL privilege to activate Bufferpool BP8K0. Compressed indexes require Bufferpools of size 8K, 16K, or 32K. Define and activate other Bufferpool names according to site requirements. For sites using Db2 data sharing, group Bufferpools must be defined with the same names.
- Access to work file database or TEMP database for Declared Global Temporary table.

SQLite authorization

Creating an SQLite database requires authority to perform the following tasks:

- Allocate, format, and mount a zFS file system
- Grant access to z/OS OMVS groups

Checklist of installation and customization tasks

This checklist includes a set of procedures for installing the product, creating a test database, populating data, and validating the test installation. When you complete these procedures, you are ready to create a production environment for IBM Z Software Asset Management.

Table 2. Checklist of installation and customization tasks

Step	Description	Data sets and members
1	Install target libraries. A z/OS system programmer performs this task. Installing target libraries	hsi = IBM Z Software Asset Management product prefix • hsi.SHSIANL1 • hsi.SHSIANL2 • hsi.SHSIEXEC • hsi.SHSIGKB1 • hsi.SHSIMJPN • hsi.SHSIMOD1 • hsi.SHSIPARM • hsi.SHSIPAX1 • hsi.SHSIPROC • hsi.SHSISAMP
2	Prepare database prerequisites. A Db2 database administrator performs this task. “Preparing Db2 database prerequisites” on page 19 The SQLite database is embedded in IBM Z Software Asset Management and the prerequisites are already configured. If you plan to use the SQLite database for your implementation, you do not have to perform this task.	Db2 SDSNSAMP data set members: • DSNTIJTM • DSNTIJCL

Table 2. Checklist of installation and customization tasks (continued)		
Step	**Description**	**Data sets and members**
3	Prepare local environment settings. Tasks include editing the HSISCUST member in the SHSISAMP target library, changing the SYSIN DD entry for local settings, and running the HSISCUST job. This job generates JCL jobs that you run in subsequent tasks. A database administrator and an IBM Z Software Asset Management administrator perform this task. Preparing local environment settings	HSIINST = HLQ for JCLLIB, and PARMLIB libraries hsi.SHSISAMP data set member: HSISCUST generates HSIINST.&DB.JCLLIB HSIINST.&DB.PARMLIB
4	Create a test Repository database. A database administrator and an IBM Z Software Asset Management administrator perform this task. • "Creating a test Repository database in Db2" on page 23 • "Creating a test Repository database in SQLite " on page 24	JCLLIB data set member: • HSISDB01 • HSISDB02 • HSISDB03 • HSISDB04 • HSISGKBL • HSISGRNT
5	Collect data and import it into the test Repository database. An IBM Z Software Asset Management administrator performs this task. "Populating the test Repository database with data" on page 24	JCLLIB data set members: • HSISINQZ: Gather Inquisitor data • HSISINQU: Gather Inquisitor UNIX data • HSISUMON: Gather Usage Monitor data • HSISIQIM: Import Inquisitor data • HSISUIMP: Import usage data

Table 2. Checklist of installation and customization tasks (continued)

Step	Description	Data sets and members
6	Create the production Repository database and arrange for regular maintenance. • "Creating a production Repository database" on page 25 • "Maintaining the production Repository database" on page 28	JCLLIB data set members: • HSISCUST • HSISDB01 • HSISDB02 • HSISDB03 • HSISDB04 • HSISGKBL • HSISGRNT • HSISGRTB • HSIJMON • HSIASALC • HSIAUTO • HSIJANLO • HSISANS1 • HSISANS2 • HSISANS3 • HSISINQZ • HSISINQU • HSISUMON • HSISIQIM • HSISUIMP • HSISUDEL • HSISLDEL • HSISPDEL • HSISMDEL • HSISTPRM • HSISUN91 • HSISLO91 • HSISUT01 • HSISUT02 • HSISUT03 • HSISUT04 • HSISIPVD • HSISAPIP • HSISLICI • HSISLICV PARMLIB data set members: • HSISMNPM • HSIAPARM • HSISANP1

Installing target libraries

Before you can install IBM Z Software Asset Management in a production environment, you can create a test environment.

Before you begin

The installation must be performed by a z/OS system programmer who has access to ShopzSeries to download the product.

Procedure

1. Download IBM Z Software Asset Management 9.1, and all available maintenance components.
2. Follow the Receive and Apply instructions in the IBM Z Software Asset Management *Program Directory* to install the target libraries.

   The following libraries are installed:

Data set low-level qualifier (LLQ)	Description
SHSIANL1	Analyzer reports.
SHSIANL2	Java script charting code.
SHSIEXEC	REXX code.
SHSIGKB1	Global Knowledge base data.
SHSIMJPN	Message templates in Japanese.
SHSIMOD1	Load modules.
SHSISAMP	Templates that the HSISCUST job uses to populate the &HSIINST..JCLLIB library.
SHSIPARM	Templates that the HSISCUST job uses to populate &HSIINST..PARMLIB library.
SHSIPROC	JCL PROCs.
SHSISAMP	Templates that the HSISCUST job uses to populate the &HSIINST..JCLLIB library.

   Note: SHSISPAX1 (REST API) is installed for IBM Z Software Asset Management.
3. Install all PTF maintenance packages.
4. Ensure that the target libraries are available to the LPAR where you intend to configure the test Db2 for z/OS database.
5. Specify that the SHSIMOD1 data set is authorized by the Authorized Program Facility (APF).
   For example, you can enter:

   ```
   SETPROG APF,ADD,DSN=hsi.SHSIMOD1,SMS
   ```

   or

   ```
   SETPROG APF,ADD,DSN=hsi.SHSIMOD1,VOL=xxxxxx
   ```
6. Schedule a change request to roll out target libraries to all z/OS LPARs where IBM Z Software Asset Management is used and include APF authorization for SHSIMOD1.
   For example, update the appropriate PROG*xx* member.

Preparing Db2 database prerequisites

The Db2 environment for the test z/OS installation has prerequisites that you must configure.

Before you begin

Db2 database administrator and IBM Z Software Asset Management administrator privileges are required to perform this task.

Db2 for z/OS must be installed. See “Installation prerequisites” on page 12. Db2 must have access to a minimum of 1600 cylinders of 3390 DASD space.

All Db2 table spaces defined in IBM Z Software Asset Management are partition-by-growth universal table spaces (UTS). UTS is a combination of partitioned and segmented table space schemes. Non-UTS table space types are deprecated.

Procedure

1. Run the DSNTIJCL job from Db2 SDSNSAMP to bind the DSNACLI plan and enable the Call Library Interface (CLI/ODBC) Db2 plan.

   If you encounter a SQL error, code -805, rebind this plan with the latest Db2 maintenance package and include the following package in the job:

   ```
   BIND PACKAGE (DSNAOCLI) MEMBER(DSNCLIMS) - CURRENTDATA(YES) ENCODING(EBCDIC)
   SQLERROR(CONTINUE)
   ```

2. Run the DSNTIJTM job from Db2 SDSNSAMP to bind the DSNREXX plan and enable the REXX Db2 plan.

Preparing local environment settings

After installation, you can create a custom version of any job in the JCLLIB library or any parameter in the PARMLIB library, by copying and editing the relevant job in the HSISCUST member in the hsi.SHSISAMP data set.

Depending on your environment, you can define parameters for the following environments:

- Db2
- SQLite
- Remote configuration

The **DBTYPE** parameter determines the environment and creates the jobs to customize and run the product in that environment.

Review the HSISCUST job parameters before you begin. A database administrator and a system programmer are required to perform the customization. After you make the required changes, submit the job. The JCL creates or reuses two output PDSE libraries and two sequential data sets.

The job creates the following PDSE libraries:

- The JCLLIB library contains Job Control Language (JCL) scripts that implement and operate the product.
- The PARMLIB library contains predefined parameters that the JCL scripts reference.

The job creates the following sequential datasets:

- The UM.HLQIDS sequential data set is referenced by the Usage Monitor on creation and contains a single record.
- The FZVSAMLOCK sequential data set is a dummy file used for serialization.

General parameters

The following table lists the general parameters that you must consider for all environments.

Table 3. General customization parameters	
Parameter	**Description**
SET HSI	You must set this JCL parameter to the high-level qualifiers of the target libraries created by the SMP/E installation process. The default parameter is hsi.V910.
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. The default parameter begins with ISP.
DBTYPE	This parameter determines the environment and creates the JCL and parameters for that environment: • Db2 • SQLITE • REMOTE: The product collects Inquisitor and Usage Monitor data at remote sites and no database is required.

Required settings for all database types

The following table lists the required settings for all databases.

Table 4. Required settings for all databases	
Parameter	**Description**
CLASS	CLASS
MSGCLASS	JES message class
MSGLEVEL	JES message level
CEERUN	This parameter specifies the fully qualified Language Environment CEERUN data set.
CBCDLL	This parameter specifies the fully qualified Language Environment CBCDLL C++ runtime data set.
HSIINST	This parameter specifies the high-level qualifier of the JCLLIB, PARMLIB, and sequential data sets that are created by running the HSISCUST job. If the JCLLIB and PARMLIB data sets exist, they are reused and members are replaced. The sequential data sets are either created or reused. The name specified for this parameter must be less than, or equal to, 19 characters in length. **Note:** For SQLite only. If you are migrating from a previous version of IBM Z Software Asset Management to 9.1, use the same value specified for this parameter. If you use a different value, a new zFS file is created and the migration will fail.

Settings for Db2 and SQLite databases

The following table lists the settings for Db2 and SQLite databases.

Table 5. Settings for Db2 and SQLite databases	
Parameter	**Description**
SYS	System where database resides

Table 5. Settings for Db2 and SQLite databases (continued)	
Parameter	**Description**
REPZSCHM	This parameter is used as a full qualifier for the tables and index definitions in the repository, and as a part qualifier for the tables and index definitions in the local knowledge base, and local knowledge base for z/OS UNIX. The REPZSCHM name must be less than, or equal to, 8 characters in length. If you are migrating from an earlier version of IBM Z Software Asset Management, use the same value specified for this parameter. If you use a different value, the migration will fail.
GKBZSCHM	This parameter is part of the table qualifier and the index definitions qualifier for the GKB, GKB for z/OS UNIX, and Inquisitor filters. The GKBZSCHM name must be less than, or equal to, 8 characters in length. If you are migrating from an earlier version of IBM Z Software Asset Management, use the same value specified for this parameter. If you use a different value, the migration will fail.
NOTFSCHM	This parameter is used as a full qualifier for the tables and index definitions in the Notifications database, and as a part qualifier for the tables and index definitions in the NOTFSCH database schema and DBNOT database. The NOTFSCHM name must be less than or equal to 8 characters in length. This parameter is new in version V9.1
DBADMIN	DBADMIN is an optional parameter. For a Db2 database, this parameter specifies the list of user IDs that are granted administrator access to the database and its contents. Specify an empty string if you do not want to grant administrator access to user IDs for the database specified in DB and DBGKB. For SQLite, this parameter specifies the list of user IDs that can connect to the z/OS RACF group.
SIZE	This parameter specifies the initial space allocation for Db2 and SQLite table spaces of the three largest tables. The default value of SIZE is 1.

Db2 database settings

The following table lists the Db2 database settings.

Table 6. Db2 database settings	
Parameter	**Description**
DB	This parameter specifies the name of the repository database that the product uses to store all of the information that it gathers other than from the GKB. The DB name must be less than, or equal to, 8 characters in length. If you are migrating from an earlier version of IBM Z Software Asset Management, use the same value specified for this parameter. If you use a different value, the migration will fail.
DBGKB	This parameter defines a single GKB database that is accessed by multiple repositories under the same Db2 subsystem. The DBGKB name must be less than, or equal to, 8 characters in length, and must not have the same name as the name defined for DB. If you are migrating from an earlier version of IBM Z Software Asset Management, use the same value specified for this parameter. If you use a different value, the migration will fail.

Table 6. Db2 database settings (continued)

Parameter	Description
DBNOT	This parameter specifies the name of the Notifications database that the product uses to store all of the notifications that it gathers from the product IBM Z Software Asset Management. The DB name must be less than, or equal to, 8 characters in length. This parameter is new in version V9.1.
DB2LOAD	This parameter specifies the fully qualified SDSNLOAD data set name.
DB2EXIT	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.
DBSSID	This parameter specifies the Db2 subsystem ID on the z/OS system.
LOC	This parameter specifies the CLI/ODBC location for the Db2 subsystem ID on the z/OS system. You can use the DB2® DISPLAY DDF command to determine the location.
SETSQLID	This parameter is used in SET CURRENT SQLID to allow a different user to define Db2 objects. This parameter is optional. The SETSQLID value must be less than, or equal to, 8 characters in length.
SGHSITAB	This parameter specifies the storage group name for small tables in the database. The default value is SGHSITAB (same as the parameter name). Consult your Db2 database administrator for security implications and naming conventions. See the SQL statement CREATE STOGROUP for more information.
SGHSIBIG	This parameter specifies the storage group name for large tables in the database. The default value is SGHSIBIG (same as the parameter name). Consult your Db2 database administrator for security implications and naming conventions. See the SQL statement CREATE STOGROUP for more information.
SGHSIIDX	This parameter specifies the storage group name for indexes in the database. The default value is SGHSIIDX (same as the parameter name). Consult your Db2 database administrator for security implications and naming conventions. See the SQL statement CREATE STOGROUP for more information.
SGTABCAT	This parameter specifies the VCAT 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. This parameter is referenced by storage group name parameter SGHSITAB.
SGTABVOL	This parameter specifies the names of the volumes that the table space data sets for small tables are allocated on. This parameter is referenced by storage group name parameter SGHSITAB.
SGBIGCAT	This parameter specifies the VCAT 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. This parameter is referenced by storage group name parameter SGHSIBIG.
SGBIGVOL	This parameter specifies the names of the volumes that the table space data sets for large tables are allocated on. This parameter is referenced by storage group name parameter SGHSIBIG.
SGIDXCAT	This parameter specifies the VCAT of the Db2 data set names for indexes in the database. Consult your Db2 database administrator for security implications and disk storage requirements. This parameter is referenced by storage group name parameter SGHSIIDX.

Table 6. Db2 database settings (continued)	
Parameter	**Description**
SGIDXVOL	This parameter specifies the names of the volumes that the data sets, for indexes, are allocated on. This parameter is referenced by storage group name parameter SGHSIIDX.
• BPDB • BPTS • BPIX	These parameters specify the buffer pool definitions for the database, table spaces, and indexes. For BPIX, the buffer pool size must be defined as 8K, 16K, or 32K.

SQLite database settings

The following table lists SQLite database settings.

Table 7. SQLite database settings	
Parameter	**Description**
SQLTZFS	The name of a zFS linear VSAM data set that is used for IBM Z Software Asset Management SQLite databases. If you are migrating from an earlier version of IBM Z Software Asset Management to 9.1 , use the same value specified for this parameter. If you use a different value, the migration will fail.
SQLTPATH	z/OS UNIX System Services directory where the SQLTZFS data set is mounted. The HSISDB01 job in JCLLIB creates this path at a later time. If you are migrating from an earlier version of IBM Z Software Asset Management to 9.1, use the same value specified for this parameter. If you use a different value, the migration will fail.

Configuring a test Repository database

Configuring the test Repository database includes setting up database objects, creating the Global Knowledge Base (GKB) database, and configuring access to the Repository and the GKB databases. Most sites maintain both a test Repository database and a production Repository database. After you configure and validate the test Repository database, repeat this task to create the production Repository database.

Creating a test Repository database in Db2

Creating the IBM Z Software Asset Management database includes setting up storage groups, the database name, and the administrator logon details. You also create the Global Knowledge Base (GKB) environment and then grant access to the database.

Procedure

1. Run the HSISDB01 job to create the storage groups.
2. Run the HSISDB02 job to create the database objects for the Global Knowledge Base (GKB).
3. Run the HSISDB03 job to create the Repository database and database objects.
4. Run the HSISDB04 job to create the Notifications database and database objects.

   Note: Before running this job, review the comments in the job. In member HSISSQ18 of the PARMLIB library, the COMPRESS keyword is set for all indexes. Bufferpool BP8K0 must first be activated in Db2® before these compressed indexes can be created. Rename and activate different Bufferpool names according to site requirements.

5. Run the HSISGKBL job to load the GKB.
6. Run the HSISGRNT job to grant DBADMIN access to the IBM Z Software Asset Management administrator to the Repository and the GKB databases.

Creating a test Repository database in SQLite

Creating the test Repository database includes allocating, formatting, and mounting a zFS file system and also granting access to z/OS OMVS groups. This is followed by creating the database name and administrator logon details.

Procedure

1. Run the HSISDB01 job to allocate, format, and mount a zFS file system.
2. Run the HSISDB02 job to create the database objects for the Global Knowledge Base (GKB).
3. Run the HSISDB03 job to create the Repository database and database objects.
4. Run the HSISDB04 job to create the Notifications database and database objects.
5. Run the HSISGKBL job to load the GKB.
6. Run the HSISGRNT job to grant access to the z/OS OMVS groups that the IBM Z Software Asset Management administrator is a member of.

Populating the test Repository database with data

You can populate the test Repository database in stages. Begin by collecting and importing Inquisitor and Usage Monitor data on the local LPAR, and then verify that this process is successful before collecting and importing data from other LPARs.

Collecting and importing data to the test Repository database

After creating the databases and database objects, you are ready to collect Inquisitor and Usage Monitor data. You can then import the collected data into the test Repository database.

Procedure

1. Run the HSISINQZ job to scan the DASD for z/OS product modules and generate output to the DD INQPZIP output file.

   For large sites, this operation can take up to an hour. You can perform step 3 while this job is running.
2. Run the HSISINQU job to scan z/OS UNIX files and generate output to the DD INQXOUT output file.

   For large sites, this operation can take up to an hour. You can perform step 3 while this job is running.
3. To run the Usage Monitor to gather initial usage data, perform the following tasks:
   a) Run the HSISUMON job to start the Usage Monitor as a batch job.

      The Usage Monitor is typically run as a started task, but you can run it as a batch job for this test. This job runs continually until you stop it manually and most of the time this job is idle.

   b) Stop the Usage Monitor to generate the `hsiinst.UM&SMF.D*.T*` output file.
   For example, enter the following command to stop the started task:

      ```
      P HSIJMON
      ```
4. To import Inquisitor (IQ) data into the test Repository database, perform the following tasks:
   a) Verify that the HSISINQZ and HSISINQU jobs that you started in steps 1 and 2, listed above, have completed.

      If the jobs are still running, wait until they are completed.

      The output logs from these jobs provide information on the number of records collected.

   b) Run the HSISIQIM job to import the data from the INQPZIP and INQXZIP output DDs that were created by the HSISINQZ and HSISINQU jobs.

For large sites, this job can take more than two hours to run the first time. Performance is significantly faster on subsequent runs.

5. Run the HSISUIMP job to import usage data from the `hsiinst.UM&SMF.D*.T*` file.

Verifying the results of the data import with the Analyzer

After you complete the collection and import of Inquisitor and Usage Monitor data, use the Analyzer to verify that the import was successful.

Procedure

1. Review the HSISANP1 PARMLIB library settings and modify if necessary.

   These settings specify the IBM Z Software Asset Management administrator user ID and password.
2. Run the HSISANLO JCLLIB job on the test Repository database.

   This job, typically, runs continually but you can enter the `F HSISANLO,STOP` command to stop it.
3. On your PC browser, logon to the Analyzer utility with the values specified in member HSISANP1 of PARMLIB library.
4. Review the Analyzer reports to confirm that all expected products have been identified.

   If a product is missing, perform the following tasks to identify the reason a product is not included:

   - Check that the product is in the GKB and report any missing product to IBM support so that they can provide an updated GKB for the product.
   - If the product exists in the GKB, check that the product is installed on the test z/OS. If the product is not installed on the test z/OS, run the Inquisitor utility on a system where the product is installed and then import that data into the test database.

Collecting and importing data from other systems

After you verify that all components are correctly installed on the test Repository database, you can now discover and import Inquisitor and Usage Monitor data from other z/OS logical partitions (LPARs).

Procedure

1. Run the following jobs to collect Inquisitor and Usage data from other systems:
   a) Run the HSISINQZ job to scan all other LPARs and generate output to the `hsiinst.HSIPZIP.Z&SMF` data set.
   b) Run the HSISINQU job and generate output to the `hsiinst.HSIUZIP.U&SMF` file.
   c) Run the HSISUMON job to start the Usage Monitor as a batch job on the other LPARS.
2. Transfer collected data to the central site via file transfer protocol (FTP).
3. Run the following jobs to import Inquisitor and Usage data at the central site:
   a) Run the HSISIQIM job to import Inquisitor data from the `hsiinst.HSIPZIP.Z&SMF` and `hsiinst.HSIUZIP.U&SMF` data sets for each LPAR.
   b) Run the HSISUIMP job to import Usage data from the `hsiinst.UM&SMF.D*.T*` data set for each LPAR.

Configuring a production Repository database

Most implementations include a test Repository database and a production Repository database. Configuring a production Repository database involves creating the database and importing data, configuring security, and automating data collection activities.

Creating a production Repository database

The production Repository database runs on a development logical partition (LPAR), and it is not necessary to run it on a business workload LPAR. You can duplicate the content of test Repository

database to populate production Repository database without collecting and importing Inquisitor and Usage Monitor data again.

About this task

You can create the production Repository database on a Db2 or SQLite database. This procedure combines instructions for both database environments. See "Configuring a test Repository database" on page 23 for database-specific instructions.

Procedure

1. Run the HSISDB01 job.
   - For Db2, the job creates storage groups.
   - For SQLite, the job allocates the zFS file system.
2. Run the HSISDB02 job to create the database objects for the Global Knowledge Base (GKB).
3. Run the HSISDB03 job to create the Repository database and database objects.

   For Db2 databases: Before running the HSISDB03 job, review and change definitions in PARMLIB members HSISSQ17 and HSISSQ18 as needed to meet site requirements:

   - In member HSISSQ17 of PARMLIB library, use the MAXPARTITIONS parameter to increase the number of partitions for the three largest table spaces (VMODULE, VUSEMTD, and VPRODDET) according to site requirements.
   - In member HSISSQ18 of the PARMLIB library, the COMPRESS keyword is set for all indexes. Bufferpool BP8K0 must first be activated in Db2 before these compressed indexes can be created. Rename and activate different Bufferpool names according to site requirements.
4. Run the HSISDB04 job to create the Notifications database and database objects
5. Run the HSISGKBL job to load the GKB.
6. Run the HSISGRNT job.
   - For Db2, the job grants DBADMIN access to the IBM Z Software Asset Management administrator for the Repository and GKB databases.
   - For SQLite, the job grants access to the z/OS OMVS groups.
7. Optional - Run the HSISGRTB job.

   For Db2, this job grants SELECT access to database tables.
8. To populate the production Repository database, repeat the procedure for collecting and importing data that you performed to populate the test Repository database.

What to do next

Configure security for the production Repository database.

Configuring security for the production Repository database

Resource Access Control Facility (RACF) security provides authentication, authorization, and auditing control for working with z/OS systems.

Procedure

1. Define a profile in the STARTED class to associate a user ID with the HSIJMON, HSIJAUTO, and HSIJANLO started tasks.
2. Specify that user IDs have the following access permissions:
   a) READ access to `hsi**` data sets
   b) ALTER access to `hsiinst.**` data sets

What to do next

Configure the automation of data collection activities on the production Repository database.

Automating data collection and reporting activities

When you configure the Usage Monitor, the Automation Server, and the Analyzer to run as started tasks, these data collection and reporting activities are automated.

Procedure

1. Configure the Usage Monitor utility to start automatically:
   a) In the HSISMNPM member of the PARMLIB data set, modify settings if necessary so that the `DSN(HSIINST.UM&SMF)` command generates `HSIINST.UM&SMF.D*.T* data sets.`
   b) Schedule a change request to roll out the new HSIJMON started task on all z/OS LPARs.
   c) Copy the HSIJMON started task from the JCLLIB library to the system PROCLIB data set.
   d) Arrange for the HSIJMON started task to start early in the initial program load (IPL) cycle to ensure that all usage activity is recorded.
2. Configure the Automation Server utility to start automatically and to automate data collection and import tasks:
   a) Schedule a change request to roll out a new HSIJAUTO started task on all z/OS LPARs.
   b) Run the HSIASALC job to define the automation control Virtual Storage Access Method (VSAM) data set.
   c) Configure the HSIAPARM settings to perform the following tasks every weekend:
      - Remote hosts: Runs an Inquisitor scan job to collect data, runs the ZCAT to amalgamate usage data, and transfers collected data via file transfer protocol (FTP).
      - Database host: Runs an Inquisitor Import job, runs a usage import job, and aggregates the data.
   d) If necessary, run the HSIASSCT job to mark existing data sets as being already processed in the automation control data set.
   e) Copy the HSIJAUTO started task from the JCLLIB library to the system PROCLIB data set.
   f) Arrange for the HSIJAUTO to start automatically at any time in the IPL cycle.
3. Configure the Analyzer utility to start automatically:
   a) Schedule a change request to roll out the new HSIJANLO started task to the production database host.
   b) Copy the HSIJANLO started task from the JCLLIB data set to the system PROCLIB data sets.
4. Configure the Analyzer utility to work with a secure socket layer (SSL) for HTTPS transport and to logon with a RACF user ID and password:
   a) In the HSISANP2 member of the PARMLIB data set, change the security parameter to SECURITY=SYSTEM,
   b) Review and edit the comments in the HSISANS1, HSISANS2, and HSISANS3 members of the JCLLIB data set to create a digital certificate that is required for SSL.
   c) Configure the HTTPPORT parameter, if you require a value other than the default value.
   d) Review the Analyzer reports to confirm that all expected products are identified.

Maintaining the production Repository database

You must perform regular maintenance tasks on the production Repository database to ensure optimal performance. The maintenance tasks cull obsolete and unwanted data and reorganize the database as necessary.

About this task

A database administrator or system programmer performs these maintenance tasks.

Procedure

1. Run the following jobs on a regular basis to delete obsolete data, save space, and improve processing time:
   a) Run the HSISUDEL job to delete usage data that are older than a specified period.
   b) Run the HSISLDEL job to delete obsolete discovery and usage data for a specified system (LPAR).
   c) Run the HSISPDEL job to physically delete data for all systems.
   d) Run the HSISMDEL job to delete obsolete audit records that are older than a specified period.
2. Run the following jobs on a regular basis to maintain the integrity and performance of data in the production Repository database:
   a) Run the HSISUT01 job to backup the Repository database in Db2 or backup the zFS file system in SQLite.
   b) Run the HSISUT02 job to restore the Repository database in Db2 or restore the zFS file system in SQLite.
   c) Run the HSISUT03 job to reorganize the Repository database in Db2.
   d) Run the HSISUT04 job to update RUNSTATS statistics for the Repository database in Db2.

Implementing deployment scenarios

Most implementations of IBM Z Software Asset Management are based on one of the common deployment scenarios. An example is provided for implementing each of these common deployment scenarios with a Db2 Repository database. You can adapt an example for use with an SQLite Repository database.

Related concepts

Planning for deployment

Before you deploy IBM Z Software Asset Management , consider which deployment option is best suited to your environment.

Scenario 1: Implementing a single Repository database with a single GKB database

The most common deployment scenario is an implementation with a single Repository database and a single global knowledge base (GKB) database.

About this task

The example deployment is for a Db2 database environment and includes the key parameters that influence this scenario.

Procedure

1. Customize an instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameters:

- **DBTYPE**=DB2
- **REPZSCHM**=REPHLQ1
- **GKBZSCHM**=GKBHLQ1
- **NOTFSCHM**=NOTHLQ1
- **DB**=REPDB1
- **DBGKB**=GKBDB1
- **DBNOT**=NOTDB1

2. Submit the HSISCUST job.
3. Create the Repository and GKB databases and grant access to them:
   a) Run the HSISDB01 job to create storage groups.
   b) Run the HSISDB02 job to create the GKB database and database objects.
   c) Run the HSISDB03 job to create the Repository database and database objects.
   d) Run the HSISDB04 job to create the Notifications database and database objects.
   e) Run the HSISGKBL job to load GKB data.
   f) Run the HSISGRNT job to grant DBADMIN access to IBM Z Software Asset Management administrator.
   g) Optional - Run the HSISGRTB job to grant SELECT access to database tables.
4. Collect Inquisitor and Usage Monitor data:
   a) Run the HSISINQZ job on all z/OS LPARs to collect Inquisitor data.
   b) Run the HSISINQU job on all z/OS LPARs to collect Inquisitor data for UNIX.
   c) Run the HSISUMON job on all z/OS LPARs to collect usage data.
   d) Run the HSISZCATjob to condense usage data, separately for each z/OS LPAR (if there are usage data from 3 LPARs, then there should be 3 condensed output files).
5. Transfer the collected Inquisitor and condensed ZCAT usage data to the central site via file transfer protocol (FTP).
6. Import Inquisitor and Usage Monitor data at the central site:
   a) Run the HSISIQIM job to import Inquisitor data into the Repository database for each LPAR.
   b) Run the HSISUIMP job to import Usage data into the Repository database for each LPAR.

Scenario 2: Implementing multiple Repositories with a shared GKB database

This deployment scenario implements two Repositories in a single Db2 subsystem that share a single global knowledge base (GKB) database. The advantage of sharing the same GKB is that you need only apply monthly updates to a single GKB database.

About this task

The example deployment is for two Repositories in the same Db2 subsystem to enable the Analyzer to browse both Repositories at the same time.

Procedure

1. Customize an instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameters.
   - **DBTYPE**=DB2
   - **REPZSCHM**=REPHLQ1
   - **GKBZSCHM**=GKBHLQ1

- **NOTFSCHM**=NOTHLQ1
- **DB**=REPDB1
- **DBGKB**=GKBDB1
- **DBNOT**=NOTDB1

2. Submit the HSISCUST job.
3. Create the Repository and GKB database and grant access to them:
   a) Run the HSISDB01 job to create storage groups.
   b) Run the HSISDB02 job to create the GKB database and database objects.
   c) Run the HSISDB03 job to create the Repository database and database objects.
   d) Run the HSISDB04 job to create the Notifications database and database objects.
   e) Run the HSISGKBL job to load GKB data.
   f) Run the HSISGRNT job to grant DBADMIN access to IBM Z Software Asset Management administrator.
   g) Optional - Run the HSISGRTB job to grant SELECT access to database tables.
4. Collect Inquisitor and Usage Monitor data to:
   a) Run the HSISINQZ job on all z/OS LPARs to collect Inquisitor data.
   b) Run the HSISINQU job on all z/OS LPARs to collect Inquisitor data for UNIX.
   c) Run the HSISUMON job on all z/OS LPARs to collect usage data.
   d) Run the HSISZCAT job to condense usage data, separately for each z/OS LPAR (if there are usage data from 3 LPARs, then there should be 3 condensed output files).
5. Transfer the collected Inquisitor and condensed ZCAT usage data to the central site via file transfer protocol (FTP).
6. Import Inquisitor and Usage Monitor data at the central site:
   a) Run the HSISIQIM job to import Inquisitor data into the Repository database for each LPAR.
   b) Run the HSISUIMP job to import Usage data into the Repository database for each LPAR.
7. Customize another instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameters:
   - **DBTYPE**=DB2
   - **REPZSCHM**=REPHLQ2
   - **GKBZSCHM**=GKBHLQ1
   - **NOTFSCHM**=NOTHLQ1
   - **DB**=REPDB2
   - **DBGKB**=GKBDB1
   - **DBNOT**=NOTDB1
8. Submit the HSISCUST job.
9. Create the second Repository and grant access to it:
   It is not necessary to run jobs to create and populate the GKB database in this step because the second Repository shares the GKB that was created in steps 3b and 3d.
   a) Run the HSISDB01 job to create storage groups.
   b) Run the HSISDB03 job to create the Repository database and database objects.
   c) Run the HSISGRNT job to grant DBADMIN access to IBM Z Software Asset Management administrator.
   d) Optional - Run the HSISGRTB job to grant SELECT access to database tables.
10. Collect Inquisitor and Usage Monitor data to add to the second Repository database:
    a) Run the HSISINQZ job on all z/OS LPARs to collect Inquisitor data.

b) Run the HSISINQU job on all z/OS LPARs to collect Inquisitor data for UNIX.

c) Run the HSISUMON job on all z/OS LPARs to collect usage data.

d) Run the HSISZCAT job to condense usage data, separately for each z/OS LPAR (if there are usage data from 3 LPARs, then there should be 3 condensed output files).

11. Transfer the collected Inquisitor and condensed ZCAT usage data to the central site via file transfer protocol (FTP).
12. Import Inquisitor and Usage Monitor data at the central site:
    a) Run the HSISIQIM job to import Inquisitor data into the second Repository database for each LPAR.
    b) Run the HSISUIMP job to import Usage data into the second Repository database for each LPAR.

What to do next

Repeat steps 7-12 for each additional Repository that you want to create, changing the values for the **REPZSCHM** and **DB** parameters for each new Repository.

Scenario 3: Implementing multiple Repositories with multiple GKB databases

This deployment scenario implements two Repositories in a single Db2 subsystem, each with its own global knowledge base (GKB) database. This deployment scenario is not common because you must apply monthly updates to each GKB database.

About this task

The example deployment is for two Repositories in the same Db2 subsystem to enable the Analyzer to browse both Repositories at the same time.

Procedure

1. Customize an instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameters:
   - **DBTYPE**=DB2
   - **REPZSCHM**=REPHLQ1
   - **GKBZSCHM**=GKBHLQ1
   - **NOTFSCHM**=NOTHLQ1
   - **DB**=REPDB1
   - **DBGKB**=GKBDB1
   - **DBNOT**=NOTDB1
2. Submit the HSISCUST job.
3. Create the first Repository and GKB database, and grant access to them:
   a) Run the HSISDB01 job to create storage groups.
   b) Run the HSISDB02 job to create the GKB database and database objects.
   c) Run the HSISDB03 job to create the Repository database and database objects.
   d) Run the HSISDB04 job to create the Notifications database and database objects.
   e) Run the HSISGKBL job to load GKB data.
   f) Run the HSISGRNT job to grant DBADMIN access to IBM Z Software Asset Management administrator.
   g) Optional - Run the HSISGRTB job to grant SELECT access to database tables.
4. Collect Inquisitor and Usage Monitor data:

a) Run the HSISINQZ job on all z/OS LPARs to collect Inquisitor data.

b) Run the HSISINQU job on all z/OS LPARs to collect Inquisitor data for UNIX.

c) Run the HSISUMON job on all z/OS LPARs to collect usage data.

d) Run the HSISZCAT job to condense usage data, separately for each z/OS LPAR (if there are usage data from 3 LPARs, then there should be 3 condensed output files).

5. Transfer the collected Inquisitor and condensed ZCAT usage data to the central site via file transfer protocol (FTP).
6. Import Inquisitor and Usage Monitor data at the central site:
   a) Run the HSISIQIM job to import Inquisitor data into the second Repository database for each LPAR.
   b) Run the HSISUIMP job to import Usage data into the second Repository database for each LPAR.
7. Customize another instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameters:
   - **DBTYPE**=DB2
   - **REPZSCHM**=REPHLQ2
   - **GKBZSCHM**=GKBHLQ2
   - **NOTFSCHM**=NOTHLQ2
   - **DB**=REPDB2
   - **DBGKB**=GKBDB2
   - **DBNOT**=NOTDB2
8. Submit the HSISCUST job.
9. Create the second Repository and second GKB database, and grant access to them:
   a) Run the HSISDB01 job to create storage groups.
   b) Run the HSISDB02 job to create the GKB database and database objects.
   c) Run the HSISDB03 job to create the Repository database and database objects.
   d) Run the HSISDB04 job to create the Notifications database and database objects.
   e) Run the HSISGKBL job to load GKB data.
   f) Run the HSISGRNT job to grant DBADMIN access to IBM Z Software Asset Management administrator.
   g) Optional - Run the HSISGRTB job to grant SELECT access to database tables.
10. Collect Inquisitor and Usage Monitor data for the second Repository database:
    a) Run the HSISINQZ job on all z/OS LPARs to collect Inquisitor data.
    b) Run the HSISINQU job on all z/OS LPARs to collect Inquisitor data for UNIX.
    c) Run the HSISUMON job on all z/OS LPARs to collect usage data.
    d) Run the HSISZCAT job to condense usage data, separately for each z/OS LPAR (if there are usage data from 3 LPARs, then there should be 3 condensed output files).
11. Transfer the collected Inquisitor and condensed ZCAT usage data to the central site via file transfer protocol (FTP).
12. Import Inquisitor and Usage Monitor data at the central site:
    a) Run the HSISIQIM job to import Inquisitor data into the second Repository database for each LPAR.
    b) Run the HSISUIMP job to import Usage data into the second Repository database for each LPAR.

What to do next

Repeat steps 7-12 for each additional Repository and GKB database that you want to create, changing the values for **REPZSCHM**, **GKBZSCHM**, **NOTFSCHM**, **DB**, **DBGKB**, **DBNOT** parameters for each new Repository and GKB database.

Scenario 4: Collecting and transferring Inquisitor and usage data from remote sites

This scenario extends each of the deployment scenarios to collect data from remote sites and transfer the data back to the central site for processing.

Procedure

1. At the remote site, install the target libraries.
2. Customize an instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameter:

   ```
   DBTYPE=REMOTE
   ```

3. Submit the HSISCUST job.
4. Collect Inquisitor and Usage Monitor data and transfer the files to the central site for processing:
   a) Run the HSISINQZ job on all z/OS LPARs to collect Inquisitor data.
   b) Run the HSISINQU job on all z/OS LPARs to collect Inquisitor data for UNIX.
   c) Run the HSISUMON job on all z/OS LPARs to collect usage data.
   d) Transfer the collected Inquisitor and Usage Monitor data to the central site using file transfer protocol (FTP).

Scenario 5: Implementing in a sysplex environment

This deployment scenario is for a sysplex environment, where the DASD is fully shared across all z/OS LPARs that belong to the sysplex. This special deployment is similar to the deployment scenarios 1, 2, or 3, but the implementation steps are slightly different. This approach is intended to achieve operational efficiency by processing only a single z/OS LPAR within a sysplex.

About this task

The example deployment is for a Db2 database environment and includes the key parameters that influence this scenario. For this scenario, assume that the sysplex contains four z/OS LPARs: MVSA, MVSB, MVSC, and MVSD.

Procedure

1. Customize an instance of the HSISCUST member in the hsi.SHSISAMP data set with the following parameters:
   - **DBTYPE**=DB2
   - **REPZSCHM**=REPHLQ1
   - **GKBZSCHM**=GKBHLQ1
   - **NOTFSCHM**=NOTHLQ1
   - **DB**=REPDB1
   - **DBGKB**=GKBDB1
   - **DBNOT**=NOTDB1
2. Submit the HSISCUST job.
3. Create the Repository and GKB databases and grant access to them:
   a) Run the HSISDB01 job to create storage groups.
   b) Run the HSISDB02 job to create the GKB database and database objects.
   c) Run the HSISDB03 job to create the Repository database and database objects.

d) Run the HSISDB04 job to create the Notifications database and database objects.

e) Run the HSISGKBL job to load GKB data.

f) Run the HSISGRNT job to grant DBADMIN access to IBM Z Software Asset Management administrator.

g) Optional - Run the HSISGRTB job to grant SELECT access to database tables.

4. Collect and import Inquisitor data for all z/OS® LPARs the first time:
   a) Run the HSISINQZ job on all four z/OS LPARs to collect Inquisitor data: MVSA, MVSB, MVSC, and MVSD. The default setting is PLX=N.
   b) Transfer the collected Inquisitor data to the central site via file transfer protocol (FTP).
   c) Run the HSISIQIM job to import Inquisitor data into the Repository database for each z/OS LPAR.
5. Collect and import Inquisitor data only for a single z/OS LPAR in subsequent scans.
   a) Set **PLX=Y** in the Inquisitor HSISINQZ job.
   b) Run the HSISINQZ job on any z/OS LPAR to collect Inquisitor data. The z/OS LPAR could be MVSA, MVSB, MVSC, or MVSD.
   c) Transfer the collected Inquisitor data to the central site via FTP.
   d) Run the HSISIQIM job to import Inquisitor data into the Repository database for the z/OS LPAR, where you previously ran HSISINQZ.
   e) Repeat steps a - d for any z/OS LPAR - MVSA, MVSB, MVSC, or MVSD, every time a new scan is required.

   Note: It is very important to understand that setting PLX=Y will work only if all LPARs belonging to a sysplex have DASD volumes that are 100% shared. If one LPAR contains DASD volumes that are unique to that LPAR, then PLX=Y must NOT be used. Instead PLX=N must be set and run separately for all LPARs belonging to the sysplex.
6. Collect and import Inquisitor data for UNIX for all z/OS LPARs the first time:
   a) Run the HSISINQU job on all four z/OS LPARs to collect Inquisitor data for UNIX.
   b) Transfer the collected Inquisitor data for UNIX to the central site via FTP.
   c) Run the HSISIQIM job to import Inquisitor data for UNIX into the Repository database for each z/OS LPAR.
   d) Repeat steps a - c for every z/OS® LPAR every time a new scan is required.
7. Collect and import Usage Monitor data:
   a) Run the HSISUMON job on all z/OS LPARs to collect usage data.
   b) Transfer the collected Usage Monitor data to the central site via FTP.
   c) Run the HSISUIMP job to import Usage data into the Repository database for each LPAR.

Migrating to IBM Z Software Asset Management 9.1

When you migrate to the latest version of IBM Z Software Asset Management 9.1 from an earlier version, you must convert existing data to be compatible with your new environment.

Migrating to IBM Z Software Asset Management from an earlier version

You can upgrade to 9.1 from IBM Z Software Asset Management 8.3/8.3.1, or from IBM Z Software Asset Management 8.2, on either a Db2 Repository database or an SQLite database. Migration from IBM Z Software Asset Management 8.1 is not supported.

Migrating from IBM Z Software Asset Management 8.3/8.3.1 to IBM Z Software Asset Management 9.1 (Db2® database)

When you upgrade to IBM Z Software Asset Management 9.1(Db2® database), there is porting of data within the Repository database. New Db2® objects are defined in the Repository. The existing 8.3/8.3.1 GKB database is dropped and re-created with the same database name for 9.1. Versions 8.3 and 8.3.1 have identical database objects for the repository and GKB databases. As such, migration to version 9.1 has the same set of instructions.

Before you begin

Make a backup of your 8.3/8.3.1 Repository database by running job HSISUT01 from your 8.3/8.3.1 JCLLIB, or equivalent in-house backup job.

Make a backup or rename your JCLLIB and PARMLIB data sets.

Migration planning and consideration:

1. If your existing 8.3/8.3.1 Repository database (including LKB/LKU) and GKB use different schema names, then you need to modify the migration jobs to suit your site requirements.
2. If you have multiple 8.3/8.3.1 repositories sharing the same 8.3/8.3.1 GKB database, you need to phase the migration process:
   - A 9.1 repository must use a 9.1 GKB.
   - An 8.3/8.3.1 repository must use an 8.3/8.3.1 GKB.

     Note: : IBM Z Software Asset Management 9.1 code fails when accessing an 8.3/8.3.1GKB database, due to newly defined 9.1 objects.

     For example, if you have 8.3/8.3.1 repositories REP1, REP2, REP3, and REP4 sharing the same 8.3/8.3.1 GKB, then migrate as follows:

     – To migrate the first repository, REP1, in 9.1 customization job HSISCUST, create a new 9.1 GKB database. For example, GKB91:

       - Customize parameters DBGKB and GKBZSCHM with different names from those used in 8.3/8.3.1 GKB database. Repository parameter settings and values remain the same.
       - Migrate REP1 and GKB database (GKB91) to 9.1 at the same time. Operational jobs for repository REP1 must now reference the 9.1 load library hsi.SHSIMOD1.
       - The remaining REPs continue to run at 8.3/8.3.1, with operational jobs still referencing the 8.3/8.3.1 load library hsi.SHSIMOD1. These REPs continue to use the 8.3/8.3.1 GKB database (GKB83).
       - Gradually migrate the remaining three repositories without creating another 9.1 GKB (GKB91).
       - Once all REPs are migrated and are using the 9.1 GKB database (GKB91), the 8.3/8.3.1 GKB (GKB83) database can be dropped.
3. If each repository has its own GKB, then migrate the Repository database, GKB database, and hsi.SHSIMOD1, to 9.1, all at the same time.
4. **Housekeeping:**

   Perform housekeeping on the 8.3/8.3.1 Repository database before you start your migration process. This reduces the time required to migrate all the data.

   a. **HSISLDEL** - If you have any obsolete LPARs in the repository, you should delete the obsolete LPARs by running HSISLDEL.

   b. **HSISPDEL** - TMODULE is one of the largest tables, and it contains modules of which a huge percentage are in-house programs. To delete obsolete modules, (especially in-house programs), refer to job HSISPDEL. You need to define a date range for deletion and a sample SQL statement is provided in the job to list date ranges. HSISPDEL deletes modules based on any load libraries that have been marked as deleted.

c. **HSISUDEL -** TUSEMTD is the largest table. Performing housekeeping on this table should be part of best practices. To determine the status of this table, run the following SQL statement:

```
SELECT FPERIOD, COUNT(*) FROM &RESPZSCHM.TUSEMTD GROUP BY FPERIOD ;
```

Following, in the Usage Deletion job HSISUDEL, select the date range for deletion. Follow the instructions in the job. If you have not used deletion before, delete Usage records in increments. Do this for all LPARs. Then run the SQL statement again to check the number of outstanding records in TUSEMTD.

A good guideline on the number of records to be retained is to run HSISUDEL monthly for all LPARs with a fixed set of parameter settings:

```
KEEPDETAIL=3(or 6)
KEEPAGGR=12
```

This will retain detailed Usage records for the current month and the previous 3 (or 6) months, and summarized records for the current month and the previous 12 months.

Continue to run your 8.3/8.3.1 Usage Monitor job/started task (HSISUMON/HSIJMON), but stop the Analyzer, and do not run any 8.3/8.3.1 operational jobs during the migration.

About this task

Perform these migration tasks for every Db2® Repository in your IBM Z Software Asset Management environment.

Procedure

Procedure

1. In 9.1, make a copy of the HSISCUST member in the hsi.SHSISAMP data set and modify the following parameters:
   a. Set the value of the **DBTYPE** parameter to DB2.
   b. Set **HSIINST** to a different value to the one defined for the existing 8.3/8.3.1 system. This will ensure that the JCLLIB/PARMLIB datasets are created with different names. As stated in the section, “Before you begin”, backup or rename copies of 8.3/8.3.1 JCLLIB/PARMLIB datasets.
   c. Set the value of the **SYS** parameter to the same system that is defined for your existing 8.3/8.3.1 Repository database.
   d. Set the value of the **DB** parameter to the same value that is defined for your existing 8.3/8.3.1 Repository database.
   e. Set the value of the **DBGKB** parameter to the same value that is defined for your existing 8.3/8.3.1 Global Knowledge Base (GKB) database.
   f. Add the value of the **DBNOT** parameter.
   g. Set the value of the **REPZSCHM** parameter to the same value that is defined for your 8.3/8.3.1 Repository database.
   h. Set the value of the **GKBZSCHM** parameter to the same value that is defined for your 8.3/8.3.1 GKB database.
   i. Add the value of the **NOTFSCHM** parameter.
   j. Set values of the remaining Db2® parameters (e.g. **DBSSID, LOC**) to the same values that are defined for your 8.3/8.3.1 Repository database.
   k. The default value for the **BPIX** parameter is set to BP8K0 and must be activated before usage. Compressed indexes require Bufferpools to be defined with BP8K0-BP8K9, BP16K0-BP16K9, or BP32K-BP32K9. It cannot be BP0-BP49.
2. Submit the HSISCUST job. DO NOT share members of JCLLIB/PARMLIB between 8.3/8.3.1 and 9.1. Some member names may be the same, but the contents differ.

3. Edit and update jobs in the JCLLIB library and parameters in the PARMLIB library if there are special site requirements
4. Run the following migration jobs:
   a. **HSISMS31**– Submit this job to display the meta data of the 8.3/8.3.1 Repository, LKB and LKU objects. Verify that the number of 8.3/8.3.1 database objects match the expected result. If the expected result does not match, DO NOT proceed to the next job, HSISMS32. Investigate why there are differences. Possible reasons are described in the comments section of the job. Upon successful completion of the job, proceed to the next job, HSISMS38.

      A condition code of 0 is expected.

   b. **HSISMS38**– Submit this job to update the Repository database, define new Db2® objects, add new columns, and modify existing columns. These are required for new functions in 9.1. Upon successful completion of the job, proceed to the next job, HSISMS39.

      A condition code of 0 is expected.

   c. **HSISMS39**– Submit this job to verify database objects implemented in the previous job, HSISMS38, have been applied successfully. Upon successful completion of the job, proceed to the next job, HSISMS31.

      A condition code of 0 is expected.

   d. **HSISMS31**– Resubmit this job to display the meta data of the newly migrated 9.1 Repository LKB and LKU objects. Verify that the number of 9.1 database objects match the expected result.

      A condition code of 0 is expected.

5. **Backup 9.1**
   a. **HSISUT01**– run the 9.1 job to backup all Repository, LKB and LKU UTS.
   b. **HSISUT04**– submit this job to run RUNSTATS for the 9.1 repository.
6. **HSISDB02**– Submit this job to drop and create a new GKB database and its dependent objects.
   - If you are creating a new 9.1 GKB database with different GKBDB/GKBZSCHM names, just submit the job. This creates a new 9.1 GKB database and its dependent objects. Use this approach if you have multiple 8.3/8.3.1 repositories sharing the same 8.3/8.3.1 GKB database. See Migration planning and consideration on page 49, step 3.
   - If you are creating the 9.1 GKB database where the GKBDB/GKBZSCHM have identical names to 8.3/8.3.1, then uncomment step //*DROPGKB. This will drop the 8.3/8.3.1 GKB database and create a new 9.1 GKB database with the same GKBDB/GKBZSCHM names as 8.3/8.3.1.

     IBM Z Software Asset Management 9.1 GKB has new database objects defined in the GKB database.
   - Upon successful completion of the job, proceed to the next job.
7. Run the **HSISDB04** job to create the Notifications database and database objects.
8. Submit job HSISGKBL to populate the 9.1 GKB database. A GKB level is shipped with this migration. To download the latest GKB level, see "Updating the Global Knowledge Base" on page 65.
9. **HSISGRTB**– Optional. Submit this job to grant privileges to users that require SELECT access to newly created 9.1 tables.
10. **Recovery**
    a. If failures occur during the migration, and a recovery is required, run the 8.3/8.3.1 job HSISUT02 to recover using the backup copy created just before the start of migration.

What to do next

After migration, use the following approach to manage the implementation to the latest version:

1. You must apply the latest GKB update. This will ensure that all product identifications are up to date when you run the Inquisitor Import job, HSISIQIM. For each repository, run HSISIQIM for all

LPARs before you run any Usage Import (HSISUIMP) jobs. You can continue to use existing 8.3/8.3.1 Inquisitor fully-scanned files as inputs for the 9.1 HSISIQIM Inquisitor Import job.

2. For each repository, run the HSISIQIM job for every LPAR with setting of FULLREMATCH=Y. For performance reasons, exclude the Aggregator job step, except for the last HSISIQIM job. Please read the comments in the "Performance consideration" section of job HSISIQIM before you proceed.
3. For the last HSISIQIM job, update the Aggregator jobstep with COUNTUSAGEFULL=Y. For example:

```
//AGGR EXEC HSIJSQLE,PROG=HSICTLAG,TPARAM=HSISAGP1
//USERPARM DD *
COUNTUSAGEFULL=Y
```

   Run the last HSISIQIM job with COUNTUSAGEFULL=Y for the Aggregator jobstep.
4. After running the last HSISIQIM job, in the Aggregator jobstep, set COUNTUSAGEFULL=N (the default setting).
5. Repeat steps 1 to 4 for the next repository.
6. Before you run any Usage Import job, HSISUIMP, you must run the Inquisitor Import job, HSISIQIM, for all LPARS (as described in step 1). Failure to complete running the Inquisitor Import job (HSISIQIM) for all LPARs before you start running Usage Import (HSISUIMP) could result in errors during the Aggregator jobstep due to the product identifications not being up-to-date.
   a. For each repository, run the HSISUIMP job for every LPAR. For performance reasons, exclude the Aggregator jobstep, except for the last HSISUIMP job. Please read the comments in the "Performance consideration" section of job HSISUIMP before you proceed.
   b. For the last HSISUIMP job, update the Aggregator jobstep with COUNTUSAGEFULL=Y. For example:

```
//AGGR EXEC HSIJSQLE,PROG=HSICTLAG,TPARAM=HSISAGP1
//USERPARM DD *
COUNTUSAGEFULL=Y
```

      Run the last HSISUIMP job with COUNTUSAGEFULL=Y for the Aggregator jobstep.
   c. After running the last HSISUIMP job in the Aggregator jobstep, set COUNTUSAGEFULL=N (the default setting).
   d. Repeat steps 6a through 6c for the next repository.
7. Configure APF authorization for the 9.1 hsi.SHSIMOD1 load library.
8. You can run 9.1 operational jobs and discontinue 8.3/8.3.1 tasks once the 9.1 Inquisitor scans and Usage Monitors are ready for use.
   a. Review the settings in the Inquisitor scan jobs, before submissions:

```
HSISINQU – PACK=1 (default)
HSISINQZ – PACK=1 (default)
```

   b. Before starting HSISUMON, review parameters:

```
PARMLIB member HSISMNPM – different parameters and default values.
```

   c. HSISZCAT – different parameters and default values:

```
Parameters 'JNM=Y,UID=Y,JAC=Y' are now the default.
```

   d. HSISIQIM – parameter COUNTUSAGE = N is now the default.

Migrating from IBM Z Software Asset Management 8.3/8.3.1 to IBM Z Software Asset Management 9.1 (SQLite database)

When you upgrade to IBM Z Software Asset Management 9.1(SQLite database), there is porting of data within the Repository database. New SQLite objects are defined in the Repository. The existing 8.3/8.3.1 GKB database is dropped and re-created with the same database name for 9.1. Versions 8.3 and 8.3.1

have identical database objects for the repository and GKB databases. As such, migration to version 9.1 has the same set of instructions.

Before you begin

Make a backup of your 8.3/8.3.1 Repository database by running job HSISUT01 from your 8.3/8.3.1 JCLLIB, or equivalent in-house backup job.

Make a backup or rename your JCLLIB and PARMLIB data sets.

Migration planning and consideration:

1. If your existing 8.3/8.3.1 Repository database (including LKB/LKU) and GKB use different schema names, then you need to modify the migration jobs to suit your site requirements.
2. Each SQLite repository with its own GKB and also its own hsi.SHSIMOD1 load library should all be migrated to 9.1 at the same time. An 9.1 hsi.SHSIMOD1 load library cannot be used by an 8.3/8.3.1 repository.
3. **Housekeeping**:

   Perform housekeeping on the 8.3/8.3.1 Repository database before you start your migration process. This reduces the time required to migrate all the data.

   a. **HSISLDEL** - If you have any obsolete LPARs in the repository, you should delete the obsolete LPARs by running HSISLDEL.

   b. **HSISPDEL** - TMODULE is one of the largest tables, and it contains modules of which a huge percentage are in-house programs. To delete obsolete modules, (especially in-house programs), refer to job HSISPDEL. You need to define a date range for deletion and a sample SQL statement is provided in the job to list date ranges. HSISPDEL deletes modules based on any load libraries that have been marked as deleted.

   c. **HSISUDEL** - TUSEMTD is the largest table. Performing housekeeping on this table should be part of best practices. To determine the status of this table, run the following SQL statement:

   ```
   SELECT FPERIOD, COUNT(*) FROM &RESPZSCHM.TUSEMTD GROUP BY FPERIOD ;
   ```

   Following, in the Usage Deletion job HSISUDEL, select the date range for deletion. Follow the instructions in the job. If you have not used deletion before, delete Usage records in increments. Do this for all LPARs. Then run the SQL statement again to check the number of outstanding records in TUSEMTD.

   A good guideline on the number of records to be retained is to run HSISUDEL monthly for all LPARs with a fixed set of parameter settings:

   ```
   KEEPDETAIL=3(or 6)
   KEEPAGGR=12
   ```

   This will retain detailed Usage records for the current month and the previous 3 (or 6) months, and summarized records for the current month and the previous 12 months.
4. Continue to run your 8.3/8.3.1 Usage Monitor job/started task (HSISUMON/HSIJMON), but stop the Analyzer, and do not run any 8.3/8.3.1 operational jobs during the migration.

About this task

Perform these migration tasks for every SQLite Repository in your IBM Z Software Asset Management environment.

Procedure

Procedure

1. In 9.1, make a copy of the HSISCUST member in the hsi.SHSISAMP data set and modify the following parameters:

a) Set the value of the **DBTYPE** parameter to SQLITE.

b) Set **HSIINST** to the same value as the one defined for the existing 8.3/8.3.1 system. As stated in the section, "Before you begin", it is imperative that you either backup or rename copies of 8.3/8.3.1 JCLLIB/PARMLIB datasets.

c) Set the value of the **SYS** parameter to the same system that is defined for your existing 8.3/8.3.1 Repository database.

d) Set the value of the **REPZSCHM** parameter to the same value that is defined for your 8.3/8.3.1 Repository database.

e) Set the value of the **GKBZSCHM** parameter to the same value that is defined for your 8.3/8.3.1 GKB database.

f) Add the value of the **NOTFSCHM** parameter.

g) Set the value of the **SQLTZFS** parameter to the same value that is defined for your 8.3/8.3.1 zFS linear VSAM data set.

h) Set the value of the **SQLTPATH** parameter to same value that is defined for your 8.3/8.3.1 path of the z/OS UNIX for Systems Services directory.

2. Submit the HSISCUST job. The JCLLIB/PARMLIB datasets created use the same names as those created in 8.3/8.3.1. The same dataset names for JCLLIB/PARMLIB must be used in 9.1 because of the relationships between the high level qualifier **HSIINST** parameter and the **SQLTZFS/SQLTPATH** parameters.

   a. **SQLTZFS** = '&HSIINST..&SYS..ZFS'

   b. **SQLTPATH** = '/u/tadz/&SQLTZFS

3. Edit and update jobs in the JCLLIB library and parameters in the PARMLIB library if there are special site requirements.

4. Run the following migration jobs:

   a. **HSISMS31**– Submit this job to display the meta data of the 8.3/8.3.1 Repository, LKB and LKU objects. Verify that the number of 8.3/8.3.1 database objects match the expected result. If the expected result does not match, DO NOT proceed to the next job, HSISMS38. Investigate why there are differences. Possible reasons are described in the comments section of the job. Upon successful completion of the job, proceed to the next job, HSISMS38.

      A condition code of 0 is expected.

   b. **HSISMS38**– Submit this job to update the Repository database, define new Db2® objects, add new columns, and modify existing columns. These are required for new functions in 9.1. Upon successful completion of the job, proceed to the next job, HSISMS39.

      A condition code of 0 is expected.

   c. **HSISMS39**– Submit this job to verify database objects implemented in the previous job, HSISMS38, have been applied successfully. Upon successful completion of the job, proceed to the next job, HSISMS31.

      A condition code of 0 is expected.

   d. **HSISMS31**– Resubmit this job to display the meta data of the newly migrated 9.1 Repository LKB and LKU objects. Verify that the number of 9.1 database objects match the expected result.

      A condition code of 0 is expected.

5. **HSISDB02**– Submit this job to drop and create a new GKB database and its dependent objects. Before submitting this job, uncomment step //*DROPGKB. This will drop the 8.3/8.3.1 GKB database and create a new 9.1 GKB database. IBM Z Software Asset Management 9.1 GKB has new database objects defined in the GKB database.

   A condition code of 0 is expected.

6. Run the HSISDB04 job to create the Notifications database and database objects.

   Upon successful completion of the job, proceed to the next job.

A condition code of 0 is expected.

7. **HSISGKBL**–Submit this job to populate the newly created 9.1 GKB database.

   A GKB level is shipped with this migration. To download the latest GKB level, see Updating the Global Knowledge Base.

8. **Backup 9.1**

   a. HSISUT01– run the 9.1 job to backup the SQLite database.

9. **Recovery**

   a. If failures occur during the migration, and a recovery is required, run the 8.3/8.3.1 job, HSISUT02 to recover using the backup copy created just before the start of migration.

What to do next

See section "What to do next", from topic "Migrating from IBM Z Software Asset Management 8.3/8.3.1 to IBM Z Software Asset Management 9.1 (DB2 database)".

Migrating from IBM Z Software Asset Management 8.2 to IBM Z Software Asset Management 9.1 (Db2 database)

When you upgrade to IBM Z Software Asset Management 9.1(Db2® database), there is porting of data within the Repository database. New Db2 objects are defined in the Repository. The existing 8.2 GKB database is dropped and re-created with the same database name for 9.1.

Before you begin

Make a backup of your 8.2 Repository database by running job HSISUT01 from your 8.2 JCLLIB, or equivalent in-house backup job.

Make a backup or rename your JCLLIB and PARMLIB data sets.

Migration planning and consideration:

1. If your existing 8.2 Repository database (including LKB/LKU) and GKB use different schema names, then you need to modify the migration jobs to suit your site requirements.
2. If you have multiple 8.2 repositories sharing the same 8.2 GKB database, you need to phase the migration process.

   - A 9.1 repository must use a 9.1 GKB.
   - An 8.2 repository must use an 8.2 GKB.

   Note: IBM Z Software Asset Management 9.1 code fails when accessing an 8.2 GKB database, due to newly defined 9.1 objects.

   For example, if you have 8.2 repositories REP1, REP2, REP3, and REP4 sharing the same 8.2 GKB, then migrate as follows:

   a. To migrate the first repository, REP1, in 9.1 customization job HSISCUST, create a new 9.1 GKB database. For example, GKB91:

      – Customize parameters DBGKB and GKBZSCHM with different names from those used in 8.2 GKB database. Repository parameter settings and values remain the same.

   b. Migrate REP1 and GKB database (GKB91) to 9.1 at the same time. Operational jobs for repository REP1 must now reference the 9.1 load library hsi.SHSIMOD1.

   c. The remaining REPs continue to run at 8.2 , with operational jobs still referencing the 8.2 load library hsi.SHSIMOD1. These REPs continue to use the 8.2 GKB database (GKB82)

   d. Gradually migrate the remaining three repositories without creating another 9.1 GKB (GKB91).

   e. Once all REPs are migrated and are using the 9.1 GKB database (GKB91), the 8.2 GKB (GKB82) database can be dropped.

3. If each repository has its own GKB, then migrate the Repository database, GKB database, and hsi.SHSIMOD1, to 9.1, all at the same time.
4. **Housekeeping**:

   Perform housekeeping on the 8.2 Repository database before you start your migration process. This reduces the time required to migrate all the data.

   a. **HSISLDEL** - If you have any obsolete LPARs in the repository, you should delete the obsolete LPARs by running job **HSISLDEL**.

   b. **HSISPDEL** - TMODULE is one of the largest tables, and it contains modules of which a huge percentage are in-house programs. To delete obsolete modules, (especially in-house programs), refer to job **HSISPDEL**. You need to define a date range for deletion and a sample SQL statement is provided in the job to list date ranges. **HSISPDEL** deletes modules based on any load libraries that have been marked as deleted.

   c. **HSISUDEL**- TUSEMTD is the largest table. Performing housekeeping on this table should be part of best practices. To determine the status of this table, run the following SQL statement:

   ```
   SELECT FPERIOD, COUNT(*) FROM &RESPZSCHM.TUSEMTD GROUP BY FPERIOD ;
   ```

   Following, in the Usage Deletion job HSISUDEL, select the date range for deletion. Follow the instructions in the job. If you have not used deletion before, delete Usage records in increments. Do this for all LPARs. Then run the SQL statement again to check the number of outstanding records in TUSEMTD.

   A good guideline on the number of records to be retained is to run HSISUDEL monthly for all LPARs with a fixed set of parameter settings:

   ```
   KEEPDETAIL=3(or 6)
   KEEPAGGR=12
   ```

   This will retain detailed Usage records for the current month and the previous 3 (or 6) months, and summarized records for the current month and the previous 12 months.
5. Continue to run your 8.2 Usage Monitor job/started task (HSISUMON/HSIJMON) , but stop the Analyzer, and do not run any 8.2 operational jobs during the migration.

About this task

Perform these migration tasks for every Db2 Repository in your IBM Z Software Asset Management environment.

Procedure

1. In 9.1, make a copy of the HSISCUST member in the hsi.SHSISAMP data set and modify the following parameters:
   a) Set the value of the **DBTYPE** parameter to DB2.
   b) Set **HSIINST** to a different value to the one defined for the existing 8.2 system. This will ensure that the JCLLIB/PARMLIB datasets are created with different names. As stated in the section, “Before you begin”, backup or rename copies of 8.2 JCLLIB/PARMLIB datasets.
   c) Set the value of the **SYS** parameter to the same system that is defined for your existing 8.2 Repository database.
   d) Set the value of the **DB** parameter to the same value that is defined for your existing 8.2 Repository database.
   e) Set the value of the **DBGKB** parameter to the same value that is defined for your existing 8.2 Global Knowledge Base (GKB) database.

   Note: If you have multiple repositories sharing the same GKB database, you must use a different **DBGKB** name to create a new 9.1 GKB database. See Migration planning and consideration, step 3.

f) Add the value of the **DBNOT** parameter.

g) Set the value of the **REPZSCHM** parameter to the same value that is defined for your 8.2 Repository database.

h) Set the value of the **GKBZSCHM** parameter to the same value that is defined for your 8.2 GKB database.

Note: If you have multiple repositories sharing the same GKB database, you must use a different **GKBZSCHM** name to create a new 9.1 GKB database. See Migration planning and consideration, step 3.

i) Add the value of the **NOTFSCHM** parameter.

j) Set values of the remaining Db2 parameters (e.g. **DBSSID, LOC**) to the same values that are defined for your 8.2 Repository database.

k) The default value for the **BPIX** parameter is set to BP8K0 and must be activated before usage. Compressed indexes require Bufferpools to be defined with BP8K0-BP8K9, BP16K0-BP16K9, or BP32K-BP32K9. It cannot be BP0-BP49.

2. Submit the HSISCUST job. DO NOT share members of JCLLIB/PARMLIB between 8.2 and 9.1. Some member names may be the same, but the contents differ.
3. Edit and update jobs in the JCLLIB library and parameters in the PARMLIB library if there are special site requirements.
4. Run the following migration jobs:

a) **HSISMS31 -** Submit this job to display the meta data of the 8.2 Repository, LKB and LKU objects. Verify that the number of 8.2 database objects match the expected result. If the expected result does not match, DO NOT proceed to the next job, HSISMS32. Investigate why there are differences. Possible reasons are described in the comments section of the job. Upon successful completion of the job, proceed to the next job, HSISMS32.

A condition code of 0 is expected.

b) **HSISMS32** – Submit this job to update the Repository database, define new Db2 objects, add new columns, and modify existing columns. These are required for new functions in 9.1. Upon successful completion of the job, proceed to the next job, HSISMS35 .

A condition code of 0 is expected.

c) **HSISMS35** – Submit this job to verify database objects implemented in the previous job, HSISMS32, have been applied successfully. Upon successful completion of the job, proceed to the next job, HSISMS31.

A condition code of 0 is expected.

d) **HSISMS31**– Resubmit this job to display the meta data of the newly migrated 9.1 Repository, LKB and LKU objects. Verify that the number of 9.1 database objects match the expected result.

A condition code of 0 is expected.

5. **Backup 9.1**

a. **HSISUT01** – run the 9.1 job to backup all Repository, LKB and LKU UTS.

b. **HSISUT04 –** submit this job to run RUNSTATS for the 9.1 repository.

6. **HSISDB02** – Submit this job to drop and create a new GKB database and its dependent objects.

- If you are creating a new 9.1 GKB database with different GKBDB/GKBZSCHM names, just submit the job. This creates a new 9.1 GKB database and its dependent objects. Use this approach if you have multiple 8.2 repositories sharing the same 8.2 GKB database. See Migration planning and consideration, step 3.
- If you are creating the 9.1 GKB database where the GKBDB/GKBZSCHM have identical names to 8.2 , then uncomment step //*DR0PGKB. This will drop the 8.2 GKB database and create a new 9.1 GKB database with the same GKBDB/GKBZSCHM names as 8.2 .
- IBM Z Software Asset Management 9.1 GKB has new database objects defined in the GKB database.

- Upon successful completion of the job, proceed to the next job.
- **A condition code of 0 is expected.**

7. Run the HSISDB04 job to create the Notifications database and database objects.
   - Upon successful completion of the job, proceed to the next job.
   - **A condition code of 0 is expected.**
8. **HSISGKBL** – Submit this job to populate the newly created 9.1 GKB database.

   A GKB level is shipped with this migration. To download the latest GKB level, see "Updating the Global Knowledge Base" on page 49.

   A condition code of 0 is expected.
9. **HSISGRTB** – Optional. Submit this job to grant privileges to users that require SELECT access to newly created 9.1 tables.
10. **Recovery**
    a) If failures occur during the migration, and a recovery is required, run the 8.2 job HSISUT02, to recover using the backup copy created just before the start of migration.

What to do next

After migration, use the following approach to manage the implementation to the latest version:

1. You must apply the latest GKB update. This will ensure that all product identifications are up to date when you run the Inquisitor Import job, HSISIQIM. For each repository, run HSISIQIM for **all** LPARs before you run any Usage Import (HSISUIMP) jobs . You can continue to use existing 8.2 Inquisitor fully-scanned files as inputs for the 9.1 HSISIQIM Inquisitor Import job.

   Note: Job HSISIQIM will fail if the GKB level used is older than 12 months.
2. For each repository, run the HSISIQIM job for every LPAR with setting of FULLREMATCH=Y. For performance reasons, exclude the Aggregator job step, except for the last HSISIQIM job. Please read the comments in the "Performance consideration" section of job HSISIQIM before you proceed.
3. For the last HSISIQIM job, update the Aggregator jobstep with COUNTUSAGEFULL=Y. For example:

   ```
   //AGGR EXEC HSIJSQLE,PROG=HSICTLAG,TPARAM=HSISAGP1
   //USERPARM DD *
   COUNTUSAGEFULL=Y
   ```

   Run the last HSISIQIM job with COUNTUSAGEFULL=Y for the Aggregator jobstep.
4. After running the last HSISIQIM job, in the Aggregator jobstep, set COUNTUSAGEFULL=N (the default setting).
5. Repeat steps 1 to 4 for the next repository.
6. Before you run any Usage Import job, HSISUIMP, you must run the Inquisitor Import job, HSISIQIM, for **all** LPARS (as described in step 1). Failure to complete running the Inquisitor Import job (HSISIQIM) for all LPARs before you start running Usage Import (HSISUIMP) could result in errors during the Aggregator jobstep due to the product identifications not being up-to-date.
   a. For each repository, run the HSISUIMP job for every LPAR. For performance reasons, exclude the Aggregator jobstep, except for the last HSISUIMP job. Please read the comments in the "Performance consideration" section of job HSISUIMP before you proceed.
   b. For the last HSISUIMP job, update the Aggregator jobstep with COUNTUSAGEFULL=Y. For example:

      ```
       //AGGR EXEC HSIJSQLE,PROG=HSICTLAG,TPARAM=HSISAGP1
      //USERPARM DD *
      COUNTUSAGEFULL=Y
      ```

      Run the last HSISUIMP job with COUNTUSAGEFULL=Y for the Aggregator jobstep.
   c. After running the last HSISUIMP job in the Aggregator jobstep, set COUNTUSAGEFULL=N (the default setting).

d. Repeat steps 6a through 6c for the next repository.

7. Configure APF authorization for the 9.1 hsi.SHSIMOD1 load library.
8. You can run 9.1 operational jobs and discontinue 8.2 tasks once the 9.1 Inquisitor scans and Usage Monitors are ready for use.

   a. Review the settings in the Inquisitor scan jobs, before submissions:

      HSISINQU – PACK=1 (default)

      HSISINQZ – PACK=1 (default)

   b. Before starting HSISUMON, review parameters:

      PARMLIB member HSISMNPM – different parameters and default values.

   c. HSISZCAT – different parameters and default values:

      Parameters 'JNM=Y,UID=Y,JAC=Y' are now the default.

   d. HSISIQIM – parameter COUNTUSAGE = N is now the default.

   e. HSISUIMP – parameter COUNTUSAGE = N is now the default.

Migrating from IBM Z Software Asset Management 8.2 to IBM Z Software Asset Management 9.1 (SQLite database)

When you upgrade to IBM Z Software Asset Management 9.1(SQLite database), there is porting of data within the Repository database. New SQLite objects are defined in the Repository. The existing 8.2 GKB database is dropped and re-created with the same database name for 9.1.

Before you begin

Make a backup of your 8.2 Repository database by running job HSISUT01 from your 8.2 JCLLIB.

Make a backup or rename your JCLLIB and PARMLIB data sets. IBM Z Software Asset Management 9.1 uses the same dataset names for JCLLIB and PARMLIB.

Migration planning and consideration:

1. If your existing 8.2 Repository database (including LKB/LKU) and GKB use different schema names, then you need to modify the migration jobs to suit your site requirements.
2. Each SQLite repository with its own GKB and also its own hsi.SHSIMOD1 load library should all be migrated to 9.1 at the same time. An 9.1 hsi.SHSIMOD1 load library cannot be used by an 8.2 repository.
3. **Housekeeping:**

   Perform housekeeping on the 8.2 Repository database before you start your migration process. This reduces the time required to migrate all the data.

   a. **HSISLDEL** - If you have any obsolete LPARs in the repository, you should delete the obsolete LPARs by running job HSISLDEL.

   b. **HSISPDEL** - TMODULE is one of the largest tables containing modules of which a huge percentage are in-house programs. To delete obsolete modules (especially in-house programs), refer to job HSISPDEL. You need to define a date range for deletion, and a sample SQL statement is provided in the job to list date ranges. HSISPDEL deletes modules based on any load libraries that have been marked as deleted.

   c. **HSISUDEL** - TUSEMTD is the largest table. Performing housekeeping on this table should be part of best practices. To determine the status of this table, run the following SQL statement:

      ```
      SELECT FPERIOD, COUNT(*) FROM &RESPZSCHM.TUSEMTD GROUP BY FPERIOD ;
      ```

      Following, in the Usage Deletion job HSISUDEL, select the date range for deletion. Follow the instructions in the job. If you have not used deletion before, you should delete Usage records

in increments. Do this for all LPARs. Then run the SQL statement again to check the number of outstanding records in TUSEMTD.

A good guideline on the number of records to be retained is to run HSISUDEL monthly for all LPARs with a fixed set of parameter settings:

```
KEEPDETAIL=3 (or 6)
KEEPAGGR=12
```

This will retain detailed Usage records for the current month and the previous 3 (or 6) months, and summarized records for the current month and the previous 12 months.

4. Continue to run your Usage Monitor job/started task (HSISUMON/ HSIJMON), but stop the Analyzer, and do not run any 8.2 operational jobs during the migration.

About this task

Perform these migration tasks for every SQLite Repository in your IBM Z Software Asset Management environment.

Procedure

1. In 9.1, make a copy of the HSISCUST member in the hsi.SHSISAMP data set and modify the following parameters:
   a) Set the value of the **DBTYPE** parameter to SQLITE.
   b) Set **HSIINST** to the same value as the one defined for the existing 8.2 system. As stated in the section, "Before you begin", it is imperative that you either backup or rename copies of 8.2 JCLLIB/ PARMLIB datasets.
   c) Set the value of the **SYS** parameter to the same system that is defined for your existing 8.2 Repository database.
   d) Set the value of the **REPZSCHM** parameter to the same value that is defined for your 8.2 Repository database.
   e) Set the value of the new **GKBZSCHM** parameter to the same value that is defined for your 8.2 GKB database
   f) Add the value of the **NOTFSCHM**parameter.
   g) Set the value of the **SQLTZFS** parameter to the same value that is defined for your 8.2 zFS linear VSAM data set
   h) Set the value of the **SQLTPATH** parameter to same value that is defined for your 8.2 path of the z/OS UNIX for Systems Services directory
2. Submit the HSISCUST job. The JCLLIB/PARMLIB datasets created use the same names as those created in 8.2 . The same dataset names for JCLLIB/PARMLIB must be used in 9.1 because of the relationships between the high level qualifier HSIINST parameter and the **SQLTZFS/SQLTPATH** parameters.
   a. **SQLTZFS** = '&HSIINST..&SYS..ZFS'
   b. **SQLTPATH** = '/u/tadz/&SQLTZFS'
3. Edit and update jobs in the JCLLIB library and parameters in the PARMLIB library if there are special site requirements.
4. Run the following migration jobs:
   a) **HSISMS31** – Submit this job to display the meta data of the 8.2 Repository, LKB and LKU objects. Verify that the number of 8.2 database objects match the expected result. If the expected result does not match, DO NOT proceed to the next job, HSISMS32. Investigate the differences. Possible reasons are described in the comments section of the job. Upon successful completion of the job, proceed to the next job, HSISMS32.

   A condition code of 0 is expected.

b) **HSISMS32**– Submit this job to update the Repository database, define new Db2 objects, add new columns, and modify existing columns. These are required for new functions in 9.1. Upon successful completion of the job, proceed to the next job, HSISMS33.

A condition code of 0 is expected.

c) **HSISMS33** Submit this job to rename 8.2 tables to names suffixed with _OLD, create new tables, and copy data from the renamed tables to the newly created tables. Upon successful completion of the job, proceed to the next job, HSISMS34.

A condition code of 0 is expected.

d) **HSISMS34** – Submit this job to drop the _OLD tables created in the previous job, HSISMS33. Upon successful completion of the job, proceed to the next job, HSISMS33.

A condition code of 0 is expected.

e) **HSISMS35** – Submit this job to verify database objects implemented in the previous job, HSISMS32, have been applied successfully. Upon successful completion of the job, proceed to the next job, HSISMS31.

A condition code of 0 is expected.

f) **HSISMS31**– Resubmit this job to display the meta data of the newly migrated 9.1 Repository, LKB and LKU objects. Verify that the number of 9.1 database objects match the expected result.

A condition code of 0 is expected.

5. **HSISDB02**– Submit this job to drop and create a new GKB database and its dependent objects. Before submitting this job, uncomment step //*DROPGKB. This will drop the 8.2 GKB database and create a new 9.1 GKB database. IBM Z Software Asset Management 9.1 GKB has new database objects defined in the GKB database.

   A condition code of 0 is expected.

6. Run the HSISDB04 job to create the Notifications database and database objects.

   - Upon successful completion of the job, proceed to the next job.
   - **A condition code of 0 is expected.**

7. **HSISGKBL** – Run this job to populate the GKB database. Always use the latest GKB version which can be found in the latest GKB PTF

   A GKB level is shipped with this migration. To download the latest GKB level, see Updating the Global Knowledge Base.

   A condition code of 0 is expected.

8. **Backup 9.1**

   a) **HSISUT01** – run the 9.1 job to backup the SQLite database.

9. **Recovery**

   a) If failures occur during the migration, and a recovery is required, run the 8.2 job HSISUT02, to recover using the backup copy created just before the start of migration.

What to do next

See section "What to do next" in topic "Migrating from 8.2 to 9.1.

Post migration errors

About this task

If the migration from the previous release is incomplete, or if a wrong combination of files, jobs, or PARMLIB members are used between the two versions, then you might get the following errors:

1. **Running 9.1 HSISGKBL load job with a previous release of GKB SHSIGKB1 input file**

```
Db2:
DSNU3351 -DED1 177 14:54:04:47 DSNURWBG- INPUT FILED 'FGADATE' NOT ENTIRELY WITHIN INPUT
RECORD
```

```
SQLSTATE: S0022 Native Error Code: -206
DSNU0171 177 14:54:04:57 DSNUGBAC - UTILITY DATABASE SERVICES MEMORY EXECUTION ABENDED,
REASON=X'00E40347

SQLIte:
INSERT INTO MV91GKB3_GKB7.TVERSION(FPOVID,FVERSIONNAME,FPPNUMNAME,
FPRODUCTID,FMODCNT.FEOSDATE,FVERSIONGUID,FFEATUREGID,FFMID,FSUITE_ID.FWFM
DATE,FGADATE)VALUES(????????????)
COULD NOT DETERMINE TIMESTAMP FORMAT FOR
```

Reason:

The previous release of GKB SHSIGKB1 input file does not contain data for the new column.

Solution

a. HSISGKBL - run this job to populate with the 9.1 GKB SHSIGKB1 input file.

2. **Running 9.1 HSISIQIM (IQ Import) with a previous release of GKB**.

```
DBMS Version:       DSN13012 - 13.01.0002
Sysplex:N
zOS IQ file dated 2025-05-30-18.13.22 collected via APAR HHSI910
Inquisitor version = 09.01.00
System start header extension records are supported.
The number of system start extension records found is:  2
Host IP address = 10.134.49.170
Host IP name = PTHOMU1.prod.hclpnp.com
GKB Version = 250328
GKB architecture level for GKB: MVGKBY83_GKB7 = 1
GKB architecture level for IBM Z SW Asset Mgmt Version 9.1 must be 2 or greater
Error during Importer initialization
Number of records in MVREPY83.TMODULE    = 666525
Number of records in MVREPY83.TUSEMTD    = 0
Number of records in MVREPY83.TLIBRARY   = 9828
Number of records in MVREPY83.TLPAR      = 1

Number of explicit commits = 0
HSIC020E Inquisitor Import encountered errors. Error code = 6074
Elapsed time: 0 days 0 Hrs, 00 Mins, 01 Secs
HSIC070I A full rematch will be performed
HSIC020E Inquisitor Import encountered errors. Error code = 6074
```

Reason:

a. The GKB database structure has not been updated to V9.1 when migrating from a previous release.

Solution

a. HSISDB02 - run this job to drop the previous release of the GKB database and create a new 9.1 GKB.

b. HSISGKBL - run this job to populate with the 9.1 GKB database.

Collecting and importing data with IBM Z Software Asset Management

IBM Z Software Asset Management includes programs that collect system and usage data, import, filter, and match this data, update the Repository tables, and make the data available for review and query.

Updating the Global Knowledge Base

IBM Z Software Asset Management provides monthly updates to the Global Knowledge Base (GKB) so that you can keep your product inventory definitions up-to-date. You can also submit items to IBM support for inclusion in GKB updates.

About this task

Updates to the GKB are available from the ShopzSeries website and you can register for notifications when updates are posted. GKB update PTFs are associated with the fix category (FIXCAT) of IBM.FZVSAM.GKB which may be referenced to advantage when automating GKB updates.

Procedure

1. Go to the ShopzSeries site and order the latest GKB PTF.
2. Use SMP/E to install the PTF. Follow the instructions specified in HOLD ACTION, if any.
3. In the GKB load job, HSISGKBL, update the SET INDSN= value with the name of the library updated by SMP/E, and then submit the job.

Applying Updates to GKB Database Via Fix Central

About this task

Monthly Global Knowledge Base (GKB) updates are available on Fix Central or via Shopz PTF. After you download an updated file, you run the GKB load job to apply the updates to your environment.

To apply updates of the GKB database:

1. Login to Fix Central with a valid IBM user ID and password.
2. Specify the following values:
   - **Field Value**
   - **Product Family Z AIOPs**
   - **Product IBM Z Software Asset Management**
   - **Installed Version 9.1**
   - **Platform z/OS**
3. Display all fixes. The format of the fix is **9.1 FSAM-z0S-LV250831**.

   Note: The last six digits signify the fix level and is in **YYMMDD** format.
4. Select the most recent version of the GKB update file and download the file as binary. The name of the update file is **FSAM91KB.XMI**.
5. Upload the **FSAM91KB.XMI.** file to the mainframe to a preallocated file with the attributes **FB 80**.
6. After the file is uploaded, receive the file, **RECEIVE INDATASET(FSAM91KB.XMI).**
7. Enter **DA**(filename) when prompted for additional information.
8. Update the **SET INDSN=** value with the name of the file that you received in the GKB load job, HSISGKBL.
9. Submit the job.

Collecting scanned libraries with the Inquisitor for z/OS

The Inquisitor is a program that scans and collects information about partitioned data set (PDS) and partitioned data set extended (PDSE) program libraries. The Inquisitor Import program takes the collected data as input to form the basis of your software inventory.

About this task

The Inquisitor Import reads data from Inquisitor scans, where the data is filtered and matched to products. The filtered, matched data is then copied to the Repository tables where it can be viewed and queried by the Analyzer reporting utility. For related information,

Related concepts

"Importing Inquisitor data" on page 110

The Inquisitor Import reads data from Inquisitor scans, where the data is filtered and matched to products. The filtered, matched data is then copied to the Repository tables where it can be viewed and queried by the Analyzer reporting utility.

Running the Inquisitor program

The HSISINQZ job in the JCLLIB library performs the Inquisitor collection. This job is generated from the HSISCUST post-installation customization job.

About this task

The length of time it takes this job to run depends on the number of volumes and libraries to be scanned. Run this job during off-peak periods.

Procedure

1. In the HSISINQZ job, check the values for the following parameters and change if necessary:
   - The **ALLMSG** parameter requests both **DSNMSG** and **PGMMSG** message logging.
   - The **PLX** parameter is set to **N** (no). Run with this setting on all systems even if you later plan to use **PLX=Y** for future scheduled scans. Review information about the **PLX** parameter before you use **PLX=Y**.
   - The **PACK** parameter is set to 1 to request that zipped output is written using the fastest level of the ***deflate*** algorithm. Higher values up to 9 can be used for better compression, but they will use more CPU time. Specify **PACK=0** if use of the ***shrink*** zip algorithm is required.
   - The **LLQ** parameter is set to **Z&SMF**. You can change this value if you want to generate data sets with unique names without changing the JCL library.

   These values are set when the HSISINQZ job is created.
2. In the program parameter string, you can specify a report message level and an override to the system identifier. Use commas to separate the various settings specified within the program parameter string.
3. Run the HSISINQZ job.

PLX parameter of the Inquisitor program

The **PLX** parameter can reduce the time it takes to scan and process different systems that have completely shared DASD, and therefore identical software inventories. When you set **PLX=Y**, the Inquisitor Import detects libraries that are shared or libraries that have not changed and quickly processes scans of these shared SIDs.

Plan your Repository to receive scans of system identifiers (SIDs) containing libraries that are unique in library name and volume, except when identically-named libraries are copies or are shared. If you have libraries that are identical in library name and volume name but are intended to have different content, place these libraries in different Repositories so that they can be processed separately.

If a library with the same library name and volume name is encountered in different SID scans, the Inquisitor Import considers the first instance that it encounters on the first SID to be the base. The Inquisitor Import treats any subsequent instances on different SIDs as shared.

The locations of all SIDs for a given library are recorded, but module discovery information is only calculated and stored when the library is encountered on its base SID. This approach ensures consistency in matching if the copies are not synchronized. The approach also saves processing time when SIDs are identical. If the Inquisitor Import encounters shared libraries, it displays their names, current SIDs, and base SIDs in the log file, and reports their number at the end of the run.

If a SID is decommissioned and is no longer available for scanning, you can run the system deletion job to remove the SID and any libraries, modules, and products that are exclusively attached to the deleted SID. For shared libraries, only the record of the library that is attached to the specified SID is removed. The contents of the library are then attached to the subsequent SID in the list which then functions as the base SID.

If the **PLX=Y** option is specified during the Inquisitor run, the Inquisitor Import applies the results of the scan from the Inquisitor file to all SIDs that were previously processed and share the same sysplex ID with the SID in the Inquisitor file. An existing library is processed on its base SID but is recorded as seen on all SIDs of the sysplex. A new library is processed on the current SID which becomes its base SID and is recorded as seen on all SIDs of the sysplex.

The Inquisitor Import requires that the Inquisitor file is more recent than any Inquisitor file that it previously processed for the same SID. If you specify the **PLX=Y** option during the Inquisitor scan, the Inquisitor file must be more recent than previously processed files of all SIDs of the sysplex.

The default value for the **PLX** parameter is **N** (no). If you intend to use the **PLX**=**Y** option to save scanning time, you must scan all SIDs at least once and present all the scans to the Inquisitor Import, so that it can determine how the different SIDs are shared.

When you specify the **PLX**=**Y** option, the Inquisitor Import processes the file in the following manner:

- Treats the content of all SIDs that share the sysplex ID with the currently-scanned SID as being identical to each other.
- Applies the scan results to all SIDs of the sysplex.

Because the Inquisitor Import process does not verify that all SIDs are identical, incorrect results can occur if the SIDs of the sysplex have different content. Use the **PLX**=**Y** option only if you are sure that all SIDs of the sysplex are identical in content at the time of the scan.

Collecting SMP/E maintenance information

When SMP/E CSI data sets are accessible to the Inquisitor, it is possible to match scanned programs with certain aspects of their SMP/E maintenance level. The accuracy of this matching is highly dependent on configuration details.

Use one of the following scan request keywords to initiate the SMP/E maintenance data extraction process that best suits your configuration:

- ALLCSISCAN - use all available SMP/E CSI data sets on the system to look for matching data. For CATALOG requests, these are found by searching the catalog for VSAM clusters with names matching the **.CSI search mask. For VTOC scan requests, CSI data sets are found when scanning volumes, so CSI data sets on excluded volumes will not be examined.
- VOLCSISCAN - only use SMP/E configurations where the GLOBAL zone resides on the same DASD volume as the program library to look for matching data.
- IPLCSISCAN - a subset of VOLCSISCAN where volume-level SMP/E data matching is only performed for IPL volumes. IPL volumes are those DASD volumes with a PDS program library named SYS1.NUCLEUS.

VOLCSISCAN and IPLCSISCAN have no effect on CATALOG requests.

The following SMP/E information may be extracted:

FMID
The function SYSMOD that owns the module.

RMID
The SYSMOD which provide the last complete module replacement.

PUT level
The name of the SOURCEID of the RMID which conforms to the format of PUTnnnn or similar if the RMID SYSMOD has such a SOURCEID.

RSU level
the name of the SOURCEID of the RMID which conforms to the format of RSUnnnn or similar if the RMIS SYSMOD has such a SOURCEID.

The Inquisitor calls the SMP/E API module GIMAPI to access and extract SMP/E information. Although the extracted information is from target zones, all API calls are made via the global zone, so the Inquisitor does not have access to target zone details such as its data set name, or the volume on which it resides. This also means that relevant SOURCEIDs assigned after the APPLY of the RMID SYSMOD will not be collected.

For SMP/E data to be accessible, the CSI data sets must be cataloged and target zone index entries in the global zone must contain the correct data set name. You need to have at least READ access to the SMP/E CSI data sets.

SMP/E information for matched programs pertains to the whole program for PROGRAM elements, and to the replaced MOD for LMOD elements.

When ALLCSISCAN processing finds a DDDEF entry for the program library, all members with matching LMOD and PROGRAM elements will have the corresponding SMP/E details assigned to them.

When no DDDEF for the library is found, ALLCSISCAN processing attempts to match program bind dates with APPLY dates to determine the RMID for LMOD elements. Data related to global zones on the same volume as the program library are examined first when looking for a match. PROGRAM elements will not be matched at all when this method is used. For PDS libraries, SCANDIR requests will not collect the bind or link edit dates used in this process, so SCANPGM should be employed for this purpose where relevant.

For VOLCSISCAN and IPLCSISCAN processing, PROGRAM and LMOD SMP/E data is assigned to programs with matching names when the program library and global zone reside on the same volume.

Inquisitor program parameters and files

The Inquisitor program has mandatory and optional parameters that affect how data is collected. The program uses some mandatory files as well as some optional files.

Table 8. Parameter settings for the Inquisitor

Parameter	Description
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.
NOVSR	Specifies that volume statistics reports normally generated from VTOC-scanning requests are not to be written.
NOEAR	Specifies that the execution activity report normally present at the end of the SYSPRINT output is not to be written.
NOCSR	Specifies that the CSI scan report normally which reports SMP/E CSI scan summaries and statistics is not to be written.
NOREP	Specifies all of NOVSR, NOEAR and NOCSR in a single keyword.

Table 8. Parameter settings for the Inquisitor (continued)	
Parameter	**Description**
NOAPF	Specifies that the Inquisitor is to run in an environment which is not APF authorized.
NOHOST	Requests that the call to HSIPHOST to collect the TCPIP host name and IP address is bypassed.
OLDTAG	Specifies that tag data members created by program HSITAGP will not be ignored but should be processed so that the tag data can be imported into the LKB.
SID=	The value is up to 4 characters long and specifies the system identifier to be contained in the data output from the Inquisitor. If the SID identifier override is omitted, the system SMF identifier is used. The SID parameter setting is used when the SMF system identifier of a system is not unique. For example: SID=SYS2
PLX=	The parameter is used to identify if the Inquisitor data being collected is part of a SYSPLEX with fully shared DASD. 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.
PLEXNAME=	The value is up to 8 characters long and specifies the sysplex identifier to be contained in the data output from the Inquisitor. If the PLEXNAME identifier override is omitted, the actual sysplex name is used. The primary purpose of the PLEXNAME parameter is to provide a means for controlling the scope of sysplex-wide inventory updates.
PACK=	The value is a single decimal digit (0-9) and specifies the level of compaction that will be used to write zipped output. The value of 0 specifies that ***shrink*** will be used. Values in the 1 to 9 range specify the compaction level of ***deflate*** to be used. PACK=1 is the program default.
LLQ=	This parameter is used to specify a suffix string made up of one or more data set name qualifiers to be appended to the name of the data set allocated to the INQPZIP and/or INQPOUT DDs. Its maximum length is 44 characters. It may contain both static and dynamic system symbols, and the user symbols &SMF. (SMF system identifier) and &SYSLPAR. (LPAR name) supplied by the Inquisitor. Use the LLQ setting when you need to create uniquely named data sets without changing the JCL.

Table 9. Files used by the Inquisitor	
Filename	**Description**
SYSPRINT	A mandatory report file.
TAGREP	An optional report file that summarizes tag data collected by the Inquisitor.
SYSIN	A mandatory request input file. It processes fixed length, variable length, and undefined record formats. Records shorter than 72 bytes will be logically extended by the Inquisitor with blanks.
INQPZIP	An optional output file that contains compressed Inquisitor data. It is written using a variable length record format. You must provide DCB information to ensure optimal use of DASD space. In the case where INQPOUT is not allocated and INQPZIP processing encounters an S213-C8 abend, the Inquisitor will attempt to write uncompressed output to this file, overriding the LRECL as appropriate.

Table 9. Files used by the Inquisitor (continued)	
Filename	**Description**
INQPOUT	An optional output file that contains uncompressed Inquisitor data. It is not specified in the packaged sample, as the use of INQPZIP is preferred, due to its reduced space requirements. INQPOUT also contains variable length records. The program supplies the appropriate LRECL. By default, system determined block size is used. If you want to direct the Inquisitor output to a compressible extended-format data set, then you should use the INQPOUT file. The INQPZIP file employs update-in-place processing, which prevents the use of DFSMS compression, although you should note the automatic handling of abend S213-C8 for INQPZIP described in the row above.
MCDS	An optional file that allocates the DFHSM MCDS data set and is required if any requests contain the REMIGRATE or NOML2 operands. Further, if supplied for other requests, you can use it to avoid recalling data sets which are not load libraries. If the DFHSM MCDS is spread over more than one data set, use the DD names MCDS2, MCDS3, and MCDS4 consecutively. This allocates all the MCDS data sets in key range order.
ABRIN	An optional SYSIN file belonging to the FDRABRP utility program that 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	An optional SYSPRINT file belonging to the FDRABRP utility program that is required if any requests contain the ABRMIG or ABRARC operands. It is an output-only file and is not processed by the Inquisitor.
DASDMAP	An optional file containing logical volume mapping assignment statements.

Inquisitor program command syntax

The Inquisitor program includes SYSIN commands and optional command operands.

SYSIN commands

The Inquisitor program uses the SCANCMD, SCANDIR, SCANPGM, and SCANDEV SYSIN commands that are described in the following table.

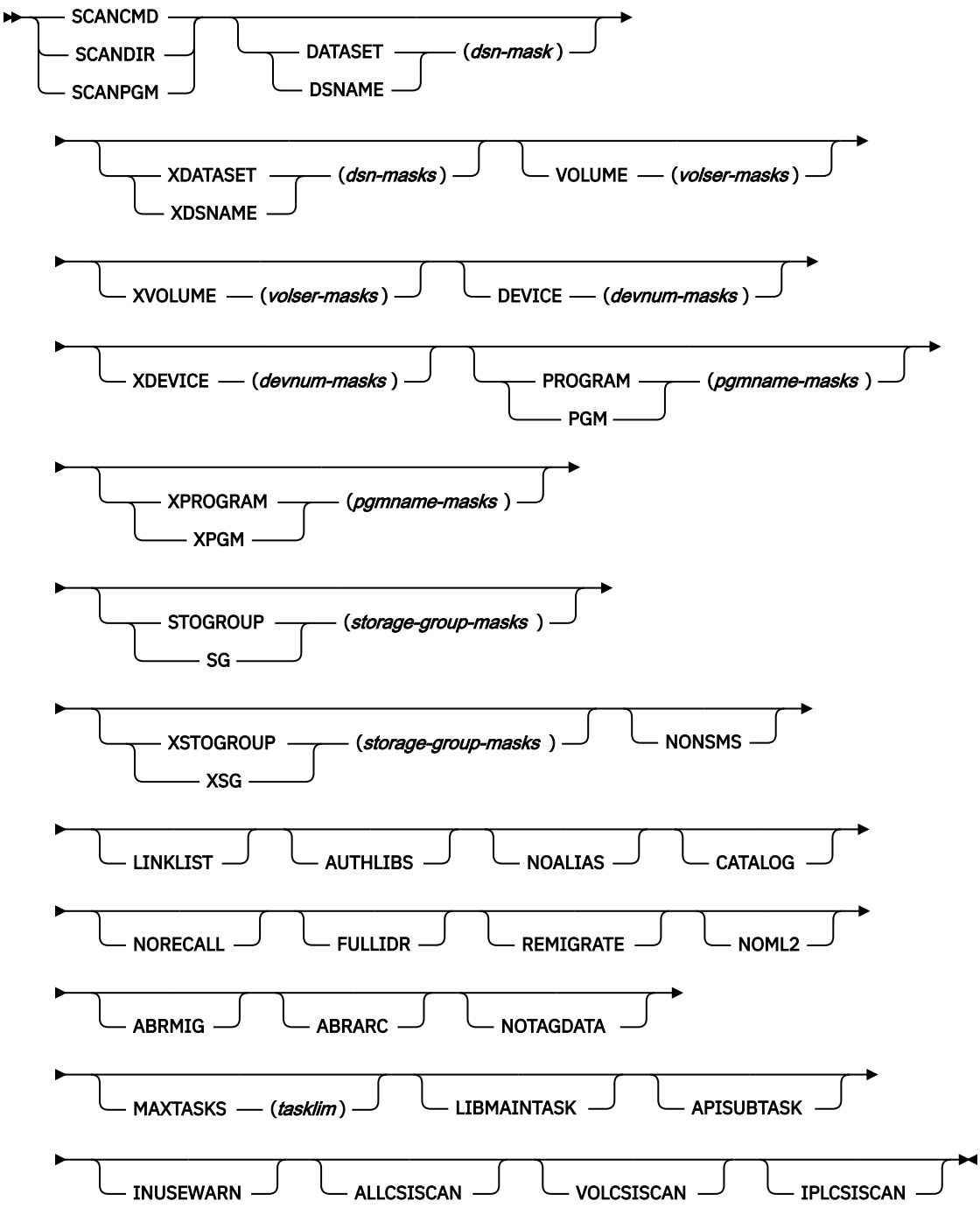
Table 10. SYSIN commands used by the Inquisitor	
Command	**Description**
SCANCMD	Allows command syntax and operand consistency to be checked by the Inquisitor without initiating an actual scan for program libraries. It performs a parse only operation, although output files are opened. Error messages relating to syntax and operand errors are produced as usual. This verb is useful if you are formulating the best request combination when implementing on any given system.
SCANDIR	Collects 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 command of choice for a production environment.

Table 10. SYSIN commands used by the Inquisitor (continued)	
Command	**Description**
SCANPGM	Collects all data collected by SCANDIR, and information from member contents. Such information relates to program structure and history. Use SCANPGM without FULLIDR if you want to collect PDS load module link edit dates and can tolerate the additional I/O and elapsed time of the scan. Use SCANPGM with FULLIDR to collect LE compiler details. This is not recommended for ongoing system-wide scans. IBM support might request SCANPGM output data to assist with problem diagnosis and resolution.
SCANDEV	Collects information about the input and output (I/O) configuration of the z/OS system including online I/O devices, control units, and related channel path connectivity. The SCANDEV command has no operands.

The Inquisitor can process multiple requests in a single program run. The output of these requests is contained in the same file.

This syntax diagram shows the SYSIN commands and their operands.

Syntax diagram of library scan commands



Operand defaults are:

```
DSNAME(*) VOLUME(*) DEVICE(*) PROGRAM(*)
```

All operands are optional. They are:

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.

DEVICE
This operand specifies one or more 1 to 4 byte device number masks. Only volumes with device numbers matching any mask specified here are processed. Volumes with device 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. Standard character string mask matching is used. The use of characters which are not hexadecimal digits will not be detected by the program.

XDEVICE
This operand specifies one or more 1 to 4 byte device number masks. Volumes with device 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 DEVICE mask. Standard character string mask matching is used. The use of characters which are not hexadecimal digits will not be detected by the program.

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.

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
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 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.

REMIGRATE
This keyword operand specifies that when a data set which had to be recalled has been processed, DFHSM 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 DFHSM 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 DFHSM 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.

MAXTASKS

This operand specifies the maximum number of VTOC-scanning subtasks to be activated by the Inquisitor. These subtasks reduce the elapsed time of an Inquisitor scan by enabling the concurrent processing of multiple volumes. Reducing the number of subtasks reduces the demand on storage in the region and does not impact performance unless the main task has to wait longer for VTOC scan results. The operand value is a decimal number in the 1 to 200 range. The default value is 10. The actual number of subtasks used does not exceed the number of volumes to be scanned. Too high a value may impact critical applications due to I/O queuing. VTOC-scanning subtasks are not used for CATALOG requests.

LIBMAINTASK

This keyword specifies that all library scan processing is to be carried out by the main task. Without this operand being specified, VTOC-scanning subtasks may scan PDS (but not PDSE) libraries for SCANDIR (but not SCANPGM) requests, which has the benefit of further reducing the elapsed time of the scan. Data collected by subtasks is held in region storage until it is processed by the main task, and SCANDIR collects a much smaller volume of data than SCANPGM. The main task uses DESERV FUNC=GET_ALL to read directories, whereas subtasks use QSAM which allows for efficient processing of corrupt PDS directories.

APISUBTASK

This keyword specifies that Program Binder API calls issued to process PDSE program objects during a SCANPGM FULLIDR request are to be performed by a subtask. Various conditions can cause Binder API calls to abend, and if such an abend occurs under the main task then Inquisitor processing is abnormally terminated. The main task can be insulated from such abends by attaching subtasks to perform the Binder API calls. Even if such subtasks abend, the main task can continue processing and should be able to complete a successful scan, provided that abending subtasks have not corrupted storage owned by the main task. Specifying this keyword will somewhat increase CPU overhead of a SCANPGM FULLIDR request, so its use is only recommended when it is known that such abends will be encountered and need to be tolerated.

INUSEWARN

This keyword specifies that failure to scan a data set because it was exclusively allocated to another job is to raise a warning condition instead of an error condition. When a data set scan attempt receives a DARC of hex 210, message HSIP092I is issued and the data set is queued for later retry.

After the scan request is complete, queued data sets are reprocessed, and one of the messages HSIP093I (successful retry), HSIP094W (failed retry with INUSEWARN) or HSIP095E (failed retry without INUSEWARN) is issued, as appropriate.

ALLCSISCAN
This keyword triggers a scan of SMP/E data sets to acquire program maintenance level information which will be associated with matched programs. For matched programs, the RMID and FMID of the PROGRAM, or most recent MOD replacement of the LMOD, will be collected. SOURCEIDs of the RMID indicating PUT or RSU level may also be collected. All detected SMP/E configurations can be searched for matching data.

VOLCSISCAN
Processing similar to ALLCSISCAN is performed, except that programs are only matched with SMP/E data accessible via global zone CSI data sets that reside on the same volume as the program library being scanned. VOLCSISCAN has no effect for CATALOG requests.

IPLCSISCAN
Processing similar to VOLCSISCAN is performed, except that SMP/E CSI data extraction is limited to program libraries residing on IPL volumes. IPL volumes have a PDS program library named SYS1.NUCLEUS. IPLCSISCAN has no effect for CATALOG requests.

SYSIN syntax rules for 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.
- Blanks and commas are equivalent.
- Subparameters of value operands are specified in parentheses.
- A continuation to the next record is requested by a plus or a hyphen when it follows a delimiter, or is at the start of a record.
- 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.
- 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.
- TSO conventions apply to abbreviations. That is, operands can be abbreviated to the minimum unambiguous length. Verbs cannot be abbreviated.
- 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.
- 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, system dynamic symbols, and &SMF (SMF system identifier) and &SYSLPAR (LPAR name), can be used to construct value masks. &SYSLPAR may resolve to a null string if z/OS is running in a virtual machine.
- 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.

Inquisitor examples

These examples show some possible scenarios where you can customize the scope and type of processing when you run the Inquisitor program.

Example 1

These three statements are equivalent, and request data collection for all programs on all online DASD volumes.

```
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*) NONSMS
```

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*)
```

Note that these 2 statements could be reduced to a single statement as follows:

```
SCANDIR VOLUME(TSO* 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 for programs with names beginning with UTIL, use: .

```
SCANDIR VOLUME(USR*) STOGROUP(SG*) PROGRAM(UTIL*)
```

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 the data in the previous example 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 linklist 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

SCANDIR DS(SYS%.**) CAT XDSN(SYS3.*,SYS4.*,SYSA.*)
```

The XDSN values are coded as shown under the assumption that SYS1, SYS2, SYS3, SYS4 and SYSA are the only 4 character high-level qualifiers beginning with SYS on the system being scanned.

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.*)
```

Designing Inquisitor requests

When constructing statements for the Inquisitor SYSIN file, try to combine all selection and exclusion criteria to form a single SCANDIR request. A single Inquisitor request will not scan a VTOC or a library more than once.

It can be difficult to formulate a system scan into a single CATALOG request, meaning that when the CATALOG operand is used, multiple requests are coded. Ensure that no data set will be scanned by more than one SCANDIR CATALOG request by excluding as many data set name patterns from each request as necessary. Data set name exclusions may not be necessary if all CATALOG search selection masks represent disjoint parts of the name space.

The example shown here uses the XDA operand to prevent SYS1.LINKLIB from being scanned more than once:

```
SCANDIR DA(SYS1.**) CATALOG
SCANDIR DA(SYS%.LINKLIB) XDA(SYS1.LINKLIB) CATALOG
```

As well as using the selection and exclusion facilities to ensure completeness, they can also be used to improve performance and efficiency by excluding DASD volumes which do not contain program libraries. Although a volume with no program libraries can be scanned quickly, processing duration might be reduced if such volumes can be excluded from an Inquisitor scan.

For example, volumes that only contain 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.

The scanning of this configuration is to be carried out with the following assumptions:

- No need for data from libraries that do not exist for more than two days.
- No program libraries on database volumes.
- Applications combine their program libraries and non-database 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, you can specify multiple requests in a single Inquisitor run. For example:

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

This can be consolidated into a single request giving the same result. For example:

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

Controlling the data extraction level

When performing a system scan, the SCANDIR verb is usually sufficient. For most libraries, SCANDIR is able to collect all the necessary information from the directory entries without the need to access member contents. For PDSE libraries, this includes collecting the bind (or link edit) date. For some specific system modules, SCANDIR will also analyze the member contents to extract additional data necessary to determine the software level of those modules.

If it is important to collect the bind dates of PDS load modules, then the SCANPGM verb can be used instead of SCANDIR. However, because SCANPGM reads multiple blocks for every member, the elapsed time required to scan each PDS library will increase by several hundred percent.

To perform maximum data extraction from every scanned program, use the SCANPGM verb with the FULLIDR keyword. This combination will greatly increase the elapsed time it takes to scan PDSE libraries as well as PDS libraries, so it is not normally expected to be used.

Extracting LE compile unit information

Specifying the FULLIDR keyword on a SCANPGM request will also allow the Inquisitor to extract information about LE compile units. Such information is stored within the program object code when a program is compiled by an LE-family compiler such as current COBOL and PL/I compilers.

The data that can be tracked for each compile unit within a scanned program includes:

- The compile date.
- The compiler level.
- The ARCH (architecture level) setting.
- The OPT (optimization level) setting.
- The DATA DIVISION statement count (for COBOL only).
- The PROCEDURE DIVISION statement count (for COBOL only).

Because of the additional resource consumption of a SCANPGM FULLIDR request over the usual SCANDIR request, it is not anticipated that a full system scan would be performed to collect this data, but rather a scan targeted to relevant application program libraries. The data from this limited scan would be used to form a purpose-built repository where data base queries can be used to extract information of interest.

When the time comes to update the system's LE compile application status, the repository would be deleted and recreated from a new targeted scan. To ascertain usage history data for these application programs, relevant queries would be directed to the main system repository where the usage data is imported.

Scanning migrated libraries

The Inquisitor locates load libraries by either scanning the VTOC of online volumes, or by searching the system catalog (CATALOG) for relevant data sets. When you use the Catalog Search Interface, you can return data sets for migrated libraries. VTOC scans do not find migrated data sets.

When the keyword CATALOG is specified in a request statement, the Inquisitor passes the data set name selection mask to the Catalog Search Interface (CSI) to search for the catalog entries. It is possible that one or more of the catalog entries returned by the CSI are for a data set that has been migrated. In contrast, VTOC scans do not find migrated data sets.

Inquisitor processing of migrated data sets found by the CSI involves dynamic allocation which then triggers the recall of the data set. Recalls increase Inquisitor processing time. The processing leaves the data set in a recalled status.

The Inquisitor looks at the volume serial number in the catalog entry to determine if a data set is migrated or not. A data set is considered to have been migrated if its catalog entry indicates a volume serial number of either MIGRAT or ARCIVE.

To suppress the processing of all migrated data sets, specify the NORECALL keyword on each Inquisitor request.

Integration with DFHSM

If you are using the MCDS file allocation, and a data set cataloged on volume MIGRAT is encountered, the Inquisitor can read the data set record from the DFHSM Migration Control Data Set (MCDS) to verify that the data has the attributes of a program library. If the MCDS record is not found, the data set is ignored and processing is bypassed, avoiding a DFHSM error condition. If the data set does not have partitioned organization, an undefined record format, and a block size of at least 1024, the Inquisitor ignores the data set, avoiding the recall of many data sets which are not program libraries.

For systems with DFHSM space management functions, you can use the request keywords NOML2 and REMIG. The MCDS file allocation is a prerequisite for using the following keywords:

NOML2
: Specifies that data sets migrated to level 2 are excluded from the scan.

REMIG
: Specifies that after a recalled data set is processed by the Inquisitor, the Inquisitor requests DFHSM to remigrate the data set. The Inquisitor does not wait for the migration to complete, but begins to process the next data set immediately after making the request to DFHSM. Migration level 2 is never specified by the Inquisitor for the migration, even if the data set was recalled from ML2. (However, it might be selected by DFHSM as a result of SMS management class settings.)

Note: Any combination of REMIGRATE, NOML2, and NORECALL is valid. Specifying NORECALL means NOML2 and REMIGRATE have no effect.

In the case where you want to scan all relevant migrated program libraries and do not want any such libraries explicitly remigrated afterward, you would not code any of the NORECALL, NOML2 and REMIGRATE keywords. In this instance, the MCDS file allocation, though optional, can still be used to great advantage.

Scanning generation data sets

Catalog Search Interface requests issued by the Inquisitor are limited to non-VSAM type A catalog entries. Generation data sets (which are members of a generation data group) are not scanned by Inquisitor CATALOG requests but can be processed by Inquisitor VTOC scans. Consider excluding generation data sets if you backup program libraries using generation data sets.

To exclude generation data sets from a VTOC scan request, specify a suitable data set exclusion mask, for example:

```
XDA(*.G%%%%V00)
```

Logical Volume Mapping

Preserving usage trends across platform upgrades

Program libraries are identified by their data set name and the volume serial number of the disk on which they reside. Program usage is attributed to these libraries in the data base. Historical usage trends can be built up over time for these software libraries. However, when a software maintenance cycle causes a program library to be replaced by a later version of the library on a different volume, or when DFHSM processing causes a program library to be relocated to another volume, the historical connection with the usage of the library on the original volume is lost.

For example, upgrading the system platform with a new set of volumes would mean that reporting the long-term usage trend for a particular program such as IDCAMS from the SYS1.LINKLIB library requires

generating several reports instead of one, even though the active SYS1.LINKLIB data set may have always resided on the IPL volume.

To counteract this, an optional facility is available where physical DASD volumes as seen by z/OS can be mapped to logical volume sets which persist beyond the time individual physical volumes are part of that logical set. With this approach, historical connections are maintained for functional program libraries which are relocated as DASD is reorganized and renewed.

The volume mapping process

The mapping is achieved by supplying simple assignment statements in a sequential file allocated using the DASDMAP file name. This file is processed by the HSIPVMAP program. HSIPVMAP is called by the Inquisitor and by the Usage Monitor writer task to translate the selected real volume serial numbers to specified 6-character strings to be considered as the volume serial numbers in data base and report processing.

The assignment statements consist of a left side, an equals sign, and a right side. Volume serial numbers encountered in the collected data which match the left side are converted to the string on the right side. The statements are processed in the order they exist in the DASDMAP file, and so the earliest matching statement will be used.

It is important that both the Inquisitor and the Usage Monitor process the same volume logical mapping assignments so that usage can be attributed to discovered inventory. The Usage Monitor writer task will process the DASDMAP file at the end of each collection cycle, so there is no need to recycle or refresh the Usage Monitor to activate updates to the file.

Real volume serial numbers are selected based on the left side of the assignment statement, and replaced in the data with the value specified on the right side of the assignment statement. Generic masking characters and system symbols may be used to generalize the mapping process so that the statements do not have to always be updated as scheduled volume reconfigurations occur.

Historical note: Users of older releases may recall the SYMVOL operand of SCANDIR, and the SYM(Y) setting of the Usage Monitor, both of which are now defunct. These allowed volume serial numbers to be represented by their symbol names. This function can be replicated using volume mapping by specifying the symbol name with two leading ampersands on the right side of an assignment statement. However, such usage of logical volume mapping is not supported unless the symbol represents the same physical volume on all relevant systems. This last restriction points to why SYMVOL and SYM(Y) were removed.

Implications for the PLX setting

The Inquisitor program HSIPINQ allows PLX=Y to be specified in its program parameter. This setting instructs Usage Import to match usage to inventory using the SYSPLEX name instead of the system identifier. This feature provides for eliminating regular Inquisitor scans on multiple systems which share the same DASD configuration. That is, after initial scans from all systems, PLX=Y allows future scans from a single system to keep the inventory up to date for all systems in the same SYSPLEX with a DASD configuration exactly matching (or entirely contained within) the DASD configuration of the primary system on which the DASD scans are to be performed on an ongoing basis.

The use of the logical volume mapping facility does not alter PLX setting requirements since PLX=Y is still permissible if all volumes are mapped to the same logical volume set on all relevant systems.

However, in cases where volume mapping depends on specific volume uses (such as IPL volume) and those uses are not the same across all relevant systems (such as the IPL volume is not shared by all relevant systems), the use of PLX=Y is not supported.

Implications of changing the volume mapping

If you add new volume mapping statements which apply only to volumes not yet processed, then this is equivalent to setting up the statements before performing any DASD scan or usage data collection, and so there is no data inconsistency. Similarly, it is safe to remove statements which apply only to volumes which have been permanently removed from the DASD subsystem.

However, introducing logical volume mapping assignment statements which change the mapping for volumes which have already been processed is equivalent to relocating the libraries on those volumes to new volumes. That is, the historical connection with previous usage of those libraries will be broken.

In such a case, usage already assigned to previously unmapped volumes will be lost. There is currently no mechanism to transfer previously accumulated usage across to the new mappings.

A data set name is unique within the scope of a single DASD volume. Logical volume mapping allows multiple data sets with the same name to appear to reside on the same DASD volume. Many-to-one mappings result in a loss of detail that cannot be exposed by subsequent reporting. This information loss may be acceptable if it improves the reportability of other information such as trends over time which may be requested by local management. Make sure you have thought out all the consequences of this before implementing such a mapping scheme.

For example, consider the scheme where all z/OS IPL volumes were labelled as RESvri where v is the version, r is the release, and i is the iteration. So, the ninth iteration of the z/OS 2.3 IPL volume is labelled RES239, and RES1DG would be the label of the sixteenth iteration of the z/OS 1.13 IPL volume.

The mapping **RES*=<Z/OS>** might be intended to preserve usage data for libraries such as SYS1.LINKLIB, but what else flows from this? For a start, data sets from all IPL volumes will appear to contain software from the latest z/OS release that is found on DASD, even though much of it will be from a different release, and possibly from a different version. This is because the latest level is always assigned when there are multiple levels that fit the data.

Mappings of **RES1*=Z/OSV1** and **RES2*=Z/OSV2** will at least allow identifying the correct program product or version, but accurate tracking of specific releases is still not possible.

Consider the following set of logical volume mapping assignments:

```
RES1D%=Z/OS1D
RES21%=Z/OS21
RES22%=Z/OS22
RES23%=Z/OS23
RES24%=z/OS24
```

This mapping set would allow each release of z/OS to be separately identified with each release's software usage being tracked. Volumes at different maintenance levels of the same release would be treated as a single volume, but this may be an acceptable loss of detail to make longer usage trends available for reporting.

Now consider grouping the usage by system rather than by OS release. The mapping **&SYSR1.=<&SMF.>** will preserve usage history for the data sets on each system's IPL volume no matter how frequently the IPL volume is cycled to a new maintenance level. As IPL volumes are switched and a new DASD scan is imported, the software levels recorded in the data base will be updated, but usage trend reports can still include data from older software levels. Such a mapping scheme could be extended to platform volumes if local volume serial naming conventions are suitable.

Bear in mind that if a program is used via a STEPLIB to a data set on an IPL volume which is not the system's current IPL volume, then that usage will be recorded against the real volume serial (assuming no other active mappings applied), but that will be correctly attributed to the software inventory from the DASD scan performed on the same system with the same set of logical volume mappings. In this scenario, each system would require a DASD scan on an ongoing basis (that is, PLX=Y would never be used) because the systems sharing the DASD would not have identical mappings.

Volume symbol mapping file

When called by the Inquisitor or the Usage Monitor, the HSIPVMAP program accesses the logical volume mapping file via the DASDMAP data definition (DD). The logical volume mapping facility is only used if the DASDMAP file allocation is present, and if the file contains valid volume symbol mapping assignment statements, and if any of these statements apply to volumes being processed. There are no specific Inquisitor or Usage Monitor parameters, settings or statement keywords which control this facility.

The assignment statements in the DASDMAP file consist of a left side character string specifying the volume serial numbers which will be replaced by the statement's operation, an equals sign, and a right side character string specifying the volume serial number to be imported into the data base.

Syntax requirements of the DASDMAP file are:

- Data from a record located after a blank or after column 72 (whichever comes first) is discarded before validation. The remaining data is referred to as *active text* in the points below.
- Because of the first point, all statements must start in column 1.
- No continuations across record boundaries are allowed.
- Active text without an equals sign is treated as a comment.
- All active text non-display characters are translated to periods. (EBCDIC code points in the x'40' to x'FE' range deemed to be display characters.)
- DBCS is not supported. DBCS characters cannot be used in volume serial numbers.
- If an ampersand is present in the active text then the system symbol substitution routine is called to perform symbol resolution.
- If the left side and/or the right side evaluate to a string longer than 6 characters then the first 6 characters will be used.
- If the left side and/or the right side evaluate to a string shorter than 6 characters then the string will be extended with blanks to a length of 6 characters.
- The first or only equals sign must be followed by a non-blank. (There is nothing to stop the use of an equals sign in the right side character string.) That is, the right side cannot be a null string.
- Generic masking characters can only be used on the left side. These characters are treated as literals if used on the right side.
- The generic masking characters are a percent sign for any single character and an asterisk for any group of zero or more characters.
- If the left side evaluates to six asterisks, it will be replaced by the current IPL volume serial number.
- Symbols are allowed in the left side and the right side character strings.
- System symbols plus the symbols &SMF (SMF system identifier) and &SYSLPAR (LPAR name) can be used. Note that &SYSLPAR may resolve to a null string if z/OS is running in a virtual machine.

Further recommendations for DASDMAP statements are:

- Do not use an ampersand character unless it is to denote the start of a symbol name.
- Do not use system dynamic symbols.
- Exploit system static symbols where appropriate so that the same set of statements can be used on multiple systems.
- Do not attempt to convert a statement into a comment by simply inserting an asterisk in column 1. Inserting a blank, or an asterisk and a blank at the start of the statement will convert it to a comment.
- Consider using a character in the right side string which is not a valid volume serial number character. This will prevent name space clashes between logical volume set names and real volume serial numbers, and also make it plain to report readers that the displayed volume is the result of a logical mapping.
- Only use EBCDIC characters in the right side string which reliably translate to invariant ASCII characters so that the string can be reliably rendered by the Analyzer in reports without any dependence on web browser language and code page settings.

The DASDMAP file may contain either fixed-length or variable-length records. Sequential concatenations of unlike data sets are supported. If you decide to use this facility, you might find it convenient to store the volume symbol mapping statements in a member of your customized PARMLIB library which was created by the HSISCUST process.

Because invalid statements are ignored, you may want to test the statements before using them in production. To do this, run the HSIPINQ program with SYSPRINT allocated to SYSOUT, SYSIN allocated to DUMMY and DASDMAP allocated to your volume mapping statement file. The absence of any output files will cause a condition code of 16, but the active mapping status will still be reported.

Volume mapping statement examples

Consider some sample statements.

```
TSO=+TSO+    /* this is a valid statement followed by a comment */
*TSO=+TSO+   so is this - text after a blank is a comment
* TSO=+TSO+  this whole line is a comment
```

The first statement would rename volume TSO to volume +TSO+ in the collected data. The second statement would rename volume serial numbers ending in TSO (such as TSO, 9TSO, 99TSO and 999TSO) to +TSO+. The third statement is a comment due to the blank in column 2 being situated before the assignment statement.

```
******=SYSRES    (6 asterisks on left treated as a special case)
*********=SYSRES (only the first 6 chars are used)
&SYSR1=SYSRES    (= and e-o-r also flag end of symbol name)
&SYSR1.=SYSRES
```

These four statements are functionally equivalent and would cause the IPL volume serial number to be replaced by the literal SYSRES.

```
WORK*=WORK*   <--- lump WORK volumes together
PUB%%%=PUB%%% <--- lump PUBLIC volumes together
```

The first statement would rename any volume serial starting with WORK to WORK* in the collected data. The second statement would rename any 6-character volume serial starting with PUB to PUB%%% in the collected data.

```
&SYSCLONEDISK=<CLON>
&SYSCLONE.DISK=<CLON>
```

where &SYSCLONE is a symbol resolving to S1.

The first statement has no effect because the literal &SYSCL cannot match any real volume serial number. The absence of a period to flag the end of the symbol name prevented any symbol substitution because the symbol &SYSCLONEDISK does not exist. The second statement will cause volume S1DISK to be reported as volume <CLON>.

```
&SYSR1=**&SMF  <--- Home system residence volume
%%%RES=*OTHRS  <--- Did we STEPLIB or JOBLIB to inactive SYSRES?
```

where all systems' IPL volumes have volume serial numbers with the fourth to sixth characters being equal to RES.

The first statement would cause the IPL volume to be reported as a volume serial number made up of two asterisks concatenated with the system identifier. Programs found to be on the IPL volumes of other systems, or perhaps on residence volumes not even in active use, would be reported as residing on volume *OTHRS (other residence volume).

```
&IMSVS=&&IMSVS  <--- Show current IMS volume as its symbol name
&DB2DA=&&DB2DA  <--- Show current DB2 volume as its symbol name
```

These two statements would cause the volumes with serial numbers which match the values of the system symbols &IMSVS and &DB2DA to be reported as &IMSVS and &DB2DA respectively.

Collecting information about the I/O configuration

The Inquisitor can scan the input and output (I/O) configuration of a z/OS system to collect information about the use of hardware assets such as storage devices. The SCANDEV command is used to request such a scan.

When you run the SCANDEV command, two additional types of records are generated which describe device groups and channel paths. Consideration of I/O devices is limited to those devices which are online at the time of the scan.

A device group is a contiguous block of device numbers, not including offline devices, where the device type, control unit type and serial number, and online channel path connectivity is the same. The channel path type is collected for each channel path used to connect to an online I/O device.

A channel that does not provide an online path to any online device is not reported even if the channel is configured online to the z/OS system.

Inquisitor completion codes

When the Inquisitor ends normally (as opposed to an abnormal end or abend) it will issue a completion code. When the Inquisitor runs as a job step, this completion code will function as a condition code that can be tested (in logic encoded in the JCL) to allow the conditional execution of subsequent job steps.

The completion codes that may be issued by the Inquisitor are:

- 0 – All processing completed successfully.
- 4 – Processing was successful but one or more warning messages (with a suffix of W) were issued reporting environmental errors which you may wish to investigate to avoid ongoing problems in application workload processing.
- 8 – Processing was successful but one or more error messages (with a suffix of E) were issued reporting encountered error conditions. It is possible that one or more warning messages were also issued.

  Generally, these errors prevented the scanning of something within the scope of the scan request. **In this case, you must assess the impact of the missing data and decide to either fix some or all of the errors and rerun the scan, or proceed with running IQ Import using the data collected by this run.**

  Another cause of completion code 8 is the failure to rename an output data set to append the low-level qualifier specified in the LLQ= parameter setting. In this case, check data set references to ensure that you process the output data you are expecting to process, and correct the data set naming problem as appropriate.
- 12 – No usable output was produced. One or more severe error messages (with a suffix of S) were issued which indicate that either no importable data was found by the scan or processing was terminated due to an invalid request or a request that was incompatible with the execution environment.
- 16 – Processing was terminated due to an unacceptable execution environment. One or more unrecoverable error messages (with a suffix of U) were issued describing the problem encountered. Missing file allocations is the most common cause of completion code 16.
- 20 – Processing was terminated due to a missing or unusable SYSPRINT allocation or a missing message module. Check the issued unrecoverable error message (with a suffix of U) to determine the cause and take remedial action.

Collecting UNIX files with the Inquisitor for z/OS UNIX

The Inquisitor for z/OS UNIX is a program that collects information about executable software existing in HFS and zFS data sets currently mounted and accessible to z/OS UNIX. The Inquisitor Import program takes the collected data as input to form the basis of your z/OS UNIX software inventory.

Inquisitor for z/OS UNIX overview

The Inquisitor for z/OS UNIX produces a set of record types which are different from those produced by the Inquisitor for z/OS. However, both programs collect the same types of information about installed software.

The Inquisitor for z/OS UNIX processes the hierarchical file system (HFS) root directory, as well as all subdirectories. For this reason, the program must run with a UID that allows access to all directories and programs 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 INQXROOT 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.

The INQXOMIT file is used to nominate one or more directories which are to be omitted or excluded from the scan, together with all of their subdirectories. This facility can be used to reduce resource consumption by preventing parts of the UNIX file hierarchy known not to have any executable software from being scanned.

Running the Inquisitor for z/OS UNIX program

The HSISINQU job in the JCLLIB library performs the Inquisitor for z/OS UNIX collection. This job is generated from the HSISCUST post-installation customization job.

About this task

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

Procedure

1. In the HSISINQU job, check the values for the following parameter and change if necessary:
   - The **PLX** parameter is set to N (no). Review information about the PLX parameter before you use PLX=Y – shared DASD is NOT sufficient to use this for a z/OS UNIX file system scan.
   - The **PACK** parameter is set to 1 to request that zipped output is written using the fastest level of the deflate algorithm. Higher values up to 9 can be used for better compression, but they will use more CPU time. Specify **PACK=0** if use of the shrink zip algorithm is required.
   - The **LLQ** parameter is set to Z&SMF. You can change this value if you want to generate data sets with unique names without changing the JCL library.This value is set when the HSISINQU job is submitted.
2. In the program parameter string, you can specify a report message level, an override to the system identifier, and whether you want compressed or uncompressed output. Use commas to separate the various settings specified within the program parameter string.

   In the program parameter string, you can specify a report message level, an override to the system identifier, and whether you want compressed or uncompressed output. Use commas to separate the various settings specified within the program parameter string.
3. Run the HSISINQU job.

PLX parameter of the z/OS® UNIX® Inquisitor program

While the earlier discussion about the PLX parameter setting for the z/OS Inquisitor program is also broadly applicable to the z/OS UNIX Inquisitor, it is important to understand that using PLX=Y for one program does not imply that PLX=Y should be used for the other program.

For the z/OS Inquisitor (which scans PDS and PDSE libraries), PLX=Y should only be specified when all the relevant systems have access to all the same libraries, and for this it is usually sufficient to share the DASD and catalog configurations.

For the z/OS UNIX Inquisitor, PLX=Y should only be specified when all the relevant systems have access to all the same UNIX files. Sharing DASD and catalogs is not sufficient to achieve this. Systems in a sysplex must be explicitly configured to share the same UNIX file system before it is suitable to use the PLX=Y setting of the UNIX Inquisitor program.

Inquisitor for z/OS UNIX program parameters and files

The Inquisitor for z/OS UNIX program has mandatory and optional parameters that affect how data is collected. The program uses some mandatory files as well as some optional files.

Table 11. Parameter settings for Inquisitor for z/OS UNIX

Parameter	Description
PTHMSG	Requests that a message is written to INQXMSG each time a directory is opened or closed.
PGMMSG	Requests that a message is written to INQXMSG each time an executable file is processed.
ALLMSG	Requests both PTHMSG and PGMMSG message logging.
NOHOST	Requests that the call to HSIPHOST to collect the TCPIP host name and IP address is bypassed.
SID=	The value is up to 4 characters long and specifies the system identifier to be contained in the data output from the Inquisitor. If the SID identifier override is omitted, the system SMF identifier is used. The SID parameter setting is used when the SMF system identifier of a system is not unique. For example: SID=SYS2
PLX=	The parameter is used to identify if the Inquisitor data being collected is from a UNIX file system shared by all systems scanned with the same sysplex name. 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.
PLEXNAME=	The value is up to 8 characters long and specifies the sysplex name to be contained in the data output from the Inquisitor. If the PLEXNAME identifier override is omitted, the actual sysplex name is used. The primary purpose of the PLEXNAME parameter is to provide a means for controlling the scope of sysplex-wide inventory updates.
PACK=	The value is a single decimal digit (0-9) and specifies the level of compaction that will be used to write zipped output. The value of 0 specifies that ***shrink*** will be used. Values in the 1 to 9 range specify the compaction level of ***deflate*** to be used. PACK=1 is the program default.
LLQ=	This parameter is used to specify a suffix string made up of one or more data set name qualifiers to be appended to the name of the data set allocated to the INQXZIP and/or INQXOUT DDs. Its maximum length is 44 characters. It may contain both static and dynamic system symbols, and the user symbols &SMF. (SMF system identifier) and &SYSLPAR. (LPAR name) supplied by the Inquisitor. Use the LLQ setting when you need to create uniquely named data sets without changing the JCL.

Table 12. Files used by the Inquisitor for z/OS UNIX

Filename	Description
INQXMSG	Report file used by HSIXINQ.
SYSPRINT	Used by Language Environment (LE), which is required to be in the standard module search path, and by IDCAMS when LLQ= is specified.
SYSOUT	Used by Language Environment (LE), which is required to be in the standard module search path.
INQXZIP	An optional output file that contains compressed Inquisitor for z/OS data. It is written using a variable length record format. You need to provide DCB information to ensure optimal use of DASD space. In the case where INQXOUT is not allocated and INQXZIP processing encounters an S213-C8 abend, the UNIX Inquisitor will attempt to write uncompressed output to this file, overriding the LRECL as appropriate.

Table 12. Files used by the Inquisitor for z/OS UNIX (continued)	
Filename	**Description**
INQXOUT	An optional output file that contains uncompressed Inquisitor for z/OS UNIX data. It is not specified in the packaged sample, as the use of INQXZIP is preferred, due to its reduced space requirements. INQXOUT also contains variable length records. The program supplies the appropriate LRECL. By default, system determined block size is used. If you want to direct the Inquisitor for z/OS UNIX output to a compressible extended-format data set, then you should use the INQXOUT file. The INQXZIP file employs update-in-place processing, which prevents the use of DFSMS compression, although you should note the automatic handling of abend S213-C8 for INQXZIP described in the row above.
INQXROOT	An optional file which can contain one or more records; each of which specifies a directory path to be considered as a root directory to be processed. If INQXROOT is empty or not allocated, then a forward slash (/) is the only root directory processed.
INQXOMIT	An optional file which can contain one or more records; each of which specifies a directory path which is to be omitted from the scan. Root directories cannot be omitted.

The INQXROOT and INQXOMIT files have the following characteristics and attributes in common:

- There is no requirement for the file to be allocated.
- The file might be empty or allocated to DUMMY.
- The file might contain fixed length or variable length records.
- Records must not contain more than 1024 bytes of data.
- Blank records are deemed to be comments and discarded.
- Leading and trailing blanks are discarded when the directory name is extracted.
- UNIX directory names may contain embedded blanks.
- The entire record is parsed so do not use sequence numbers.
- Records with an asterisk as the first non-blank are deemed to be comments and discarded.
- If the directory path does not end in a slash, then one is appended.

z/OS UNIX Inquisitor example

In this example, the requirement is for a full scan of the file system, but it is known that the DEPLOY 1 product has a vast subdirectory structure containing thousands of files, with the actual software for the product being confined to its bin subdirectory. Accordingly, to reduce unproductive processing the decision is made to exclude the product's files from the z/OS UNIX Inquisitor scan except for its bin subdirectory. The main DEPLOY 1 directory is called deploy1 and it is a subdirectory of the system's root directory.

To implement this exclusion filtering, the INQXROOT file and the INQXOMIT file contain the records as indicated below.

INQXROOT contents:

```
* System root directory
/
* DEPLOY-1 executable program directory
/deploy1/bin/
```

INQXOMIT contents:

```
* DEPLOY-1 mount-point directory
/deploy1/
```

The result is as follows:

- The z/OS UNIX Inquisitor will begin scanning the entries in the system root directory.
- As each subdirectory is discovered, the subdirectory path name will be compared to path names supplied from the INQXOMIT file and discarded if a match is found. In this way, the /deploy1 directory and all its subdirectories will be excluded from the scan.
- After the remainder of the file system has been scanned, the z/OS UNIX Inquisitor will scan the /deploy1/bin directory and its subdirectories if it has any. Again, subdirectory path names will be compared with paths named in the INQXOMIT file, but no exact matches will be found since none of the names can equal /deploy1/.

Security considerations

If you want to collect all relevant z/OS UNIX data, you must have access to all UNIX directories, including the root directory. This access ensures that all z/OS UNIX data is collected.

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

The following sample definition can 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(IBMUSER)
PE  SUPERUSER.FILESYS.** CLASS(UNIXPRIV) RESET
PE  SUPERUSER.FILESYS.** CLASS(UNIXPRIV) ID(USERONE) ACCESS(READ)
SETR CLASSACT(UNIXPRIV)
SETR RACLIST(UNIXPRIV)
SETR RACLIST(UNIXPRIV) REFR
```

Collecting usage data with the Usage Monitor

The Usage Monitor is a server address space that runs as a started task. Work is queued to the Usage Monitor from all address spaces where programs are used. The Usage Monitor moves captured program usage data into the collection repository and periodically writes the accumulated data to a sequential file. The Usage Monitor runs APF authorized and is non-swappable.

Related concepts

Importing Usage data

The Usage Import job imports data generated by the Usage Monitor and aggregates usage data for discovered or identified modules in the Repository tables in the database.

Setting up the Usage Monitor

The Usage Monitor uses the HSISUMON job in the JCLLIB library. This job is generated from the HSISCUST post-installation customization job. This job will call the HSIJMON procedure from the JCLLIB library.

The parameter for the Usage Monitor job is the HSISMNPM member in the PARMLIB library.

Files used by the Usage Monitor

The Usage Monitor has three product-specific files. They are:

UMONIN

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

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

Data set `&HSIINST..&DB..UM.HLQIDS` contains the high-level qualifier listing for products and is populated by the IQ Import job (HSISIQIM). To minimize the number of records to be created by the

Usage Monitor, only usage events that match the list of products in this data set are generated. To activate this facility, in //UMONIN, uncomment data set &HSIINST..&DB..UM.HLQIDS as described in PROC HSIJMON.

UMONMSG
A log file which contains the initial commands issued, and which indicates their degree of success. It also contains regular status reports, writer reports, refresh reports (when appropriate), and a termination report. It consists of fixed-length 121-byte records.

DASDMAP
An optional file containing logical volume mapping assignment statements.

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. This is done in the PARMLIB member HSISMNPM.

Using 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 that allow some data to be excluded from collection are provided. They are exclusion masking based on program name, and exclusion masking based on data set name.

Filtering by program name

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. Program name exclusion filtering occurs before the data set name of the program library is determined by the Usage Monitor, which makes it more efficient than data set name filtering.

Each table entry contains a program name comparison 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.

In order to add, reset, remove, or display the entries to the table, 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.

The program name exclusion table contains numerous default entries to exclude data pertaining to the usage of many programs which are part of the operating system. You can use the DEL(*ALL*) command to deactivate all default program filter entries, and the EXC(*DFLT*) command to reactivate all default program filter entries.

Unlike masks added by the EXC command, default program name exclusion masks do not exclude job step program usage events. For example, the IEF* default exclusion mask excludes dynamic calls and loads of program IEFBR14, but usages where IEFBR14 is invoked by JCL are not excluded by this mask.

While the default program exclusion list is subject to change, it can be displayed at any time by the D-X command, and is likely to include masks listed in the following table:

Program exclusion mask	System component
ATB*	APPC
ATR*	Resource recovery services

Program exclusion mask	System component
BLS*	IPCS
BPX*	z/OS UNIX system services
CBDUS*	HCD unit support
CBR*	OAM
CEA*	Common Event Adapter
CEE*	Language Environment
CEH*	Language Environment
CEJ*	Language Environment
CEL*	Language Environment
CEU*	Language Environment
CRT*	C++ Standard Library
CSR*	Callable service requests
EDC*	C/C++ Library
EZA*	Communication Server for IP Services
EZB*	Communication Server for IP Services
FLM*	SCLM
FSUM*	z/OS UNIX Shell and Utilities
HWI*	Base Control Program internal interface (BCPii)
HWT*	z/OS Client Web Enablement Toolkit
IDC*	Access Method Services (AMS)
IEA*	Supervisor control
IEC*	Device support
IED*	TSO terminal input/output controller (TIOC)
IEE*	Master Scheduler, COMMTASK, DIDOCS, etc.
IEF*	Job scheduling
IEW*	Program management
IGC*	Catalog, DADSM
IGG*	Catalog, DADSM, PAM, SAM
IKJ*	TSO TIOC, TSO/E
IKT*	TSO/VTAM
IRX*	TSO/E REXX
ISP*	ISPF
ISR*	ISPF/PDF
IST*	VTAM
IUT*	VTAM

Filtering by calling program name

For program use events resulting from LINK, LOAD and DELETE requests (but not from ATTACH requests) an additional filtering check is made against the name of the program that issued the request. This facility allows the exclusion of usage generated by software auditing and reporting tools, as well as other administrative tools which invoke programs to report whether the owning component is enabled or disabled. To state it another way, this is an indirect filtering facility where usage of the named program is not excluded, but programs fetched by the named program do have their usage excluded no matter what names those programs have.

There are no default entries for this filtering . All entries in this filter list are supplied via the LDX command. The LDD command is available to deactivate previously added entries. Exact program names (with no masking) up to 8 characters long must be specified when using LDX and LDD commands.

In order to add, remove, or display the entries in the calling program exclusion list, use these commands:

LDX
To add one or several entries to the calling program name exclusion list.

LDD
To remove one or several entries previously defined using LDX.

D-X
To display active LDX exclusion entries after default and EXC exclusion entries.

Filtering by data set name

After the Usage Monitor has ascertained the name of the data set from which a used program is fetched, it is used to decide if the usage data is retained for collection or discarded. To perform this process, three lists of data set name masks are scanned; the first is the default data set name exclusion list, the second is the dynamic data set name inclusion list, and the third is the dynamic data set name exclusion list.

The default data set name exclusion list is built during Usage Monitor initialization, and consists of the SCEERUN library, the SCEERUN2 library, SYS1.CMDLIB (containing TSO commands), SYS1.CSSLIB (containing callable services modules), and masks for data set names ending with DBDLIB and PSBLIB. The data set names of the SCEERUN and SCEERUN2 Language Environment libraries are determined by searching the link list for specific LE modules. You can use the XDD command to deactivate any of these default exclusion entries. You can use the XDS(*DFLT*) command to reactivate the default data set exclude list without affecting the status of masks in other lists.

The other two lists are constructed from commands you specify either in the UMONIN file or dynamically via the system MODIFY command.

To avoid excessive storage and processor resource consumption, 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 can be specified. If you do not supply any data set name exclusion masks, the inclusion list does not affect data collection, but can still be used as a convenient way to collect relative usage statistics from the regular Usage Monitor status reports.

Data set name filtering occurs in the following sequence:

1. Excludes usage if the data set name matches a default exclusion mask, otherwise proceeds to step 2.
2. Includes usage if the data set name matches a mask supplied by an IDS command, otherwise proceeds to step 3.
3. Excludes usage if the data set name matches a mask supplied by an XDS command, otherwise proceeds to step 4.
4. Includes usage if the data set name does not match any of the masks.

Data set name mask values are specified as a character string up to 44 bytes in length. You can use a percent sign as a wildcard to match any single character. You can use an asterisk as a wildcard to match

any group of zero or more characters. Before the mask is stored, all occurrences of *% are changed to %*, and then all occurrences of multiple consecutive asterisks are changed to a single asterisk.

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

XDS
To add a data set name mask to the exclusion list.

IDS
To add a data set name mask to the inclusion list.

XDD
To deactivate a data set name exclusion mask.

IDD
To deactivate a data set name inclusion mask.

D-D
To display all active data set name masks.

You can use the XDD(*ALL*) command to deactivate all data set exclusion masks, including those in the default list. You can use the IDD(*ALL*) command to deactivate all data set inclusion masks.

Both of the non-default lists have no elements until an XDS or IDS command is processed. The default exclusion list is provided to reduce overhead by discarding data that does not assist with product use identification, and while additional data set filtering is provided for local use, it is expected that for many systems no customization of data set filtering would be necessary.

However, if you want to take the approach of only monitoring programs from program libraries known to contain program product software, you can adopt the HLQIDS scheme that flows from the HSISCUST process which creates a UM.HLQIDS data set to be added to the UMONIN DD concatenation. UM.HLDIDS contains a XDS(*) setting to exclude all data sets followed by IDS settings with masks that include the high-level qualifiers of program libraries. You should verify that XDS(*) is not the only setting in the UM.HLQIDS data set before using it.

Filtering by UNIX pathname

If the mask value specified in an IDS, XDS, IDD, or XDD command contains at least one slash, the value is deemed to be a UNIX path name mask and not a data set name mask. During processing, multiple consecutive slashes are reduced to a single slash.

UNIX path name masks entered via IDS and XDS commands are compared to the path names specified by applications at run time and may not correspond to the path names against which usage is attributed. The main cause for this difference in path names is the use of symbolic links. The Usage Monitor writer task converts path names with symbolic links to real path names in order to match inventory discovered by the Inquisitor.

Do not use an IDD or XDD command that specifies a UNIX path name mask because the only use for such a command is to dynamically delete a UNIX path name mask. Most UNIX path name masks contain lowercase alphabetic characters. The system MODIFY command interface usually changes lower case characters to upper case which prevents the mask matching the relevant active mask. To delete a UNIX path name mask you must either recycle the Usage Monitor or use the REF command to refresh the settings from the UMONIN file. In either case, all UNIX path name masks are deactivated and the necessary change is to remove the IDS or XDS command that you want to deactivate from the UMONIN file.

Similarly, because of the prevalence of lower case alphabetics in UNIX path names, you only specify IDS and XDS commands with path name masks as UMONIN file input rather than via the MODIFY system command interface.

The length limit of 44 characters also applies to UNIX path name masks.

Recording CPU time

The Usage Monitor will attempt to record the CPU time used by each program. This is only done for task-type programs that receive control from the operating system's contents supervisor. Programs which are pre-loaded and then branched to will not accrue any recorded CPU time. Recorded CPU time for a program may include time for programs and services that it has invoked, including programs that have been excluded from collection, but will not include time attributed to other monitored programs.

Accumulated CPU time is logged in the collection repository when a monitored program ends. This means that CPU time for long-running jobs will either not be collected, or will be collected once when the component is shutdown, perhaps even with the bulk of the CPU time being due to activity in prior collection cycles, which could compromise the value of that data.

For this reason, the Usage Monitor can collect consumed CPU time differently for those address spaces which are classed as being long-running. For these address spaces, all the consumed CPU time will be attributed to the job step program, with the CPU time values being sourced from the SMF30CPT field of type 30, subtypes 2 and 3 SMF records. The previously described CPU time tracking at the request block level will not be performed for these address spaces.

Long-running address spaces are recognized by their job step program name. The initial list of long-running program names is formed from the system's program properties table at the time that the Usage Monitor is started. Additional names can be added by issuing the LRP command. Existing entries in the long-running program name list can be deleted by the LRD command. The Usage Monitor will deem an address space to be long-running if its job step program name matches any active entry in this name list. The Usage Monitor does not check whether any of the properties described by the PPT entry are actually assigned. Only started task and batch job address spaces are checked for being long-running.

This facility allows the Usage Monitor to report CPU time used by CICS, IMS control region, Db2 and MQ subsystem, and various other address space types without having to wait until those address spaces terminate.

In order to add, remove, or display the entries in the long-running program name list, use these commands:

LRP
To add one or several entries to the long-running program list.

LRD
To remove one or several entries from the long-running program list.

D-L
To display active long-running program name entries.

Starting and stopping the Usage Monitor

A Usage Monitor member named HSIJMON is provided in SHSIPROC. If you want to start HSIJMON as a started task, copy the customized member from the JCLLIB to an authorized PROCLIB.

Procedure

1. To start the Usage Monitor in normal mode, enter the following command:

```
S HSIJMON
```

2. To fully stop the Usage Monitor, enter one of the following commands:

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

These commands cause the Usage Monitor to stop data collection, attach a writer task to process the existing data in the collection repository, wait for the writer task to output the data, and then terminate.

3. To perform an immediate termination, enter the following command:

```
F HSIJMON,CAN
```

This command causes the server address space to stop data collection, detaches any running writer task which renders the output data in the data set unusable, deletes the current collection repository without writing out its contents, and terminates. If you use the z/OS system command CANCEL to stop the Usage Monitor, its collection repository remains in storage. To clear the collection repository from storage, you must restart the Usage Monitor.

Refresh processing for the Usage Monitor

The Usage Monitor includes commands that you can issue dynamically to alter processing but that are active only for the duration of the current Usage Monitor session. To implement a change to both the running Usage Monitor and to the initialization commands for starting subsequent Usage Monitor sessions, you can use the refresh facility.

Refresh processing involves the execution of the command stream placed in the UMONIN 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. This ensures that future Usage Monitor sessions do not encounter initialization command stream errors.

Some commands set a switch for logic control, or set a numeric value to be used during 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, so are part of an accumulation of commands which specify future processing.

Consider the example where several exclusion masks are active, and a change to deactivate one of the masks is required. A command to deactivate the mask might be issued dynamically, but if this change is to be made permanent, then the UMONIN file needs to be updated. The alternative is to remove the command setting the exclusion from the UMONIN file, and to then issue the Usage Monitor REF command to initiate a refresh.

Before the first UMONIN 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 UMONIN commands.

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

Stopping the Usage Monitor and restarting it, produces the same active exclusion masks as a refresh. It also produces a data collection outage. For more information, see the REF command in the next topic for a list of the processes performed during a refresh operation.

Usage Monitor general commands

Usage Monitor commands can be divided into two groups:

- General commands, which are those not in the second group.
- Trace commands, which are those related to the Usage Monitor Trace Facility.

Both general and Trace Facility commands are passed to the Usage Monitor from the UMONIN input file, or by a system MODIFY command.

The syntax rules are as follows:

- All general commands are three characters long.

- All trace-related commands are longer than three characters, and begin with TRC.
- Operands or subparameters are specified in parentheses.
- Multiple subparameters are separated by commas.
- The command must not contain any embedded blanks.
- Commands must start in column one.
- Commands setting optional values (such as a volume or an SMS class) also accept parentheses () without a subparameter to clear a previously set value.

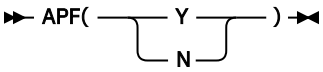
Settings are not preserved across successive runs of the Usage Monitor. Default values are set when the Usage Monitor address space initializes prior to processing the UMONIN file.

To record the settings the Usage Monitor is using, place the display commands at the end of the UMONIN file.

Details of each general command follow, with trace-related commands being described in the next section.

APF - Set APF library collection status

APF is used to specify whether output data sets written by the writer task will include a list of APF authorized libraries or not. While program HSIPINQ will collect the APF status of program libraries, it can only report that status for the system on which it executes, and it may not always be run on every system where the Usage Monitor is running. Enabling this data collection allows the data base to reflect the correct APF status of libraries for each system where the Usage Monitor is active.



Y
Specifies that APF library information is to be collected.

N
Specifies that APF library information is not to be collected.

If no APF command is issued, then APF library information is collected. APF(Y) is the default setting.

The APF setting is only referenced by the writer task at the end of a collection cycle, and the APF library information collected represents the status at that time.

ASZ - Specify ASID array area size

ASZ is used to specify the virtual storage size of the SCOPE=COMMON memory object which forms the area where the program management status of each address space is stored. The ASZ value is referenced each time a new extent of this area is allocated. The first extent is created when the Usage Monitor first initializes after an IPL. Secondary extents are allocated dynamically as required up to a maximum of 16 extents.

This area of storage is normally never freed, but the area can be deleted and a new area created by recycling the Usage Monitor with PARM='NEWASVT' specified on the job step EXEC JCL statement.

segments

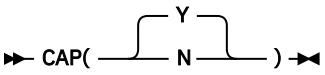
specifies a number of segments from 1 to 200.

If no ASZ command is issued, a 16 MB area will be used. As new job steps start in any given ASID, storage is reclaimed and reused so that number of pages that need to be physically backed is kept to a minimum.

CAP - Set hardware capacity collection status

CAP is used to specify if the Usage Monitor is to produce 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 collection repository switch.



Y
Specifies that hardware capacity data is collected and written out.

N
Specifies that hardware capacity is not collected or written out.

If no CAP command is issued, the default is CAP(Y).

Table 13. Examples of using the CAP command

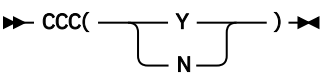
Command purpose	Example code
Collect hardware capacity data.	`F HSIJMON,CAP(Y)`
Do not collect hardware capacity data.	`F HSIJMON,CAP(N)`

CCC - Collect CICS Collection CPU time

CCC is used to control whether the Usage Monitor CICS usage collector will attempt to track the CPU time used by the Usage Monitor CICS usage collection process which measures the additional CPU time consumed by monitoring CICS usage. The time is measured from when the Usage Monitor CICS GLUE is given control by CICS until the invoked collector routine is ready to exit and return control to the GLUE. The time for each collection event is stored in storage. The latest stored value can be displayed by a D-C command. Generally, this collection is not required unless undertaking in-depth performance analysis or you are asked to enable it by IBM support.

This setting does not alter the data collected for data base import.

This setting will have no effect if there are no active CICS regions customized for Usage Monitor usage collection, or if CIC(N) is set.



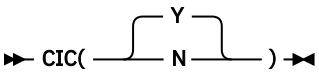
Y
Specifies that additional system service calls will be made to track a few microseconds of CPU time consumed by CICS usage collection.

N
Specifies that system service timing requests will not be made to track CICS usage collection CPU time.

CCC(N) is the default setting.

CIC - Allow or disable program usage data from CICS regions

The CIC command provides a system-wide control mechanism to allow or disallow program usage data to be collected by the Usage Monitor CICS® global user exit (GLUE) program.



Y
Specifies that customized CICS regions are able to present program usage data to the Usage Monitor for collection.

N
Specifies CICS program usage monitoring is disabled throughout the operating system image.

If no CIC command is issued, the default is CIC(Y).

D-A - Display output allocation parameters

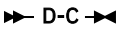
D-A is used to display dynamic allocation details to be used in the creation of output data files. The data set name, DCB attributes, primary and secondary space quantities, and unit and optional volume serial number are shown.

The following code example displays the current dynamic allocation values.

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

D-C - Display the counters and statistics

D-C is used to display 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.



The following code example displays the current value of internal Usage Monitor counters.

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

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

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

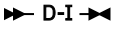
The inclusion and exclusion lists do not need to be populated in order to collect data. The absence of any entries in the exclusion list means that data collection is not filtered by program library data set names.

The following code example displays the current data set name inclusion and exclusion lists.

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

D-I - Display the system identifier

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



The following example code displays the current system identifier used by the Usage Monitor.

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

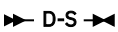
D-L - Display long-running program names

D-L is used to list the program names currently considered to be long-running. The initial list is formed from the system's program properties table active when the Usage Monitor initializes. Names can be added to the list by the LRP command, and removed from the list by the LRD command.

When the name of the job step program of a started task or batch job matches an active entry in this list, detailed CPU time consumed by programs in this step is not tracked, but the step's CPU time consumed each SMF interval is attributed to the job step program, allowing some tracking of consumed CPU time for rarely-ending address spaces.

D-S - Display the status settings

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



Place at the end of the UMONIN file to confirm monitoring settings.

The following example code displays the current values of settings.

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

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

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

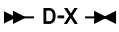
The UTC or GMT switch time is calculated using local time current at collection repository creation time. The time when a collection repository is terminated is set when it is created. Changes to the system local time offset, such as those caused by a change to daylight saving time, do not alter the UTC or GMT that the current collection repository is closed. The time of the switch after the next switch is calculated using the new local time.

The following example code displays 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.

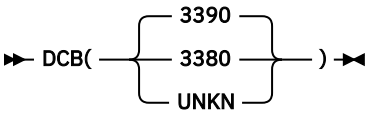


The following example code displays the current exclude list entries.

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

DCB - Set output DCB attributes

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



If no DCB command is issued, the default is DCB(3390).

DCB(3390)
Sets the output DCB to

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

Use when the output device has 3390 compatible geometry.

DCB(3380)
Sets the output DCB to

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

Use when the output device has 3380 compatible geometry.

DCB(UNKN)
Sets the output DCB to

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

The system determines the optimal block size for the device used by dynamic allocation. Use when the output device type is not known until allocation time.

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

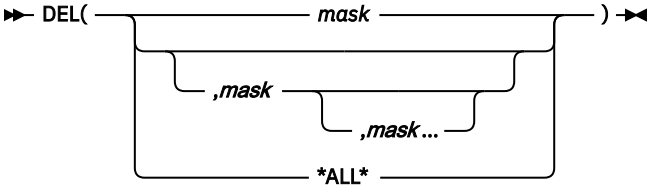
DCL - Set the data set SMS data class

The DCL command is used to override the data class that is assigned to dynamically created output data sets by the system's ACS routines. You may use this, for example, to assign a data class that is not eligible to be marked SMS compressible so that the Usage Monitor can output zipped data.

dataclas
specifies a 1 – 8 character SMS data class name

DEL - Deleting program mask entries

DEL is 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.



mask
Specifies a 1 - 8 character program name mask. Any wildcard characters in the mask are treated as literals for the purposes of finding the mask to delete.

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.

Table 14. Examples of using the DEL command

Command purpose	Example code
Remove all entries, so that all possible programs are monitored.	`F HSIJMON,DEL(*ALL*)`
Remove exclusion masks to monitor LE and REXX modules.	`F HSIJMON,DEL(CEE*,IRX*)`
Remove an exclusion mask to monitor 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. The UMONIN file must contain a DSN command.

You can use symbols in the construction of the data set name prefix. Available symbols include all z/OS static symbols, &SMF, the SMF identifier for the system, and &SYSLPAR, the logical partition name for the system.

►► DSN(*dsnpref*) ►◄

dsnpref
Specifies a 1 - 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.

Usage Monitor needs RACF ALTER access to the data sets to be able to create them.

The following example code shows how to get output files with names of the form SYS3.HSI.HSIJMON.*Dyyyyddd.Thhmmsst*:

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

DUR - Set execution duration

DUR is used to specify a fixed short-term execution duration of the Usage Monitor started task. When the specified time has elapsed the Usage Monitor will terminate automatically. The Usage Monitor stop time is calculated by adding the specified duration to the current time when the command is processed.

Any subsequent WRT commands are ignored.

The DUR command is not normally used in standard operations where the Usage Monitor is to remain active until system shutdown. When it is used, it is normally placed in the UMONIN file to specify a predetermined length of execution for sampling or testing purposes.

►► DUR(*hhmm*) ►◄

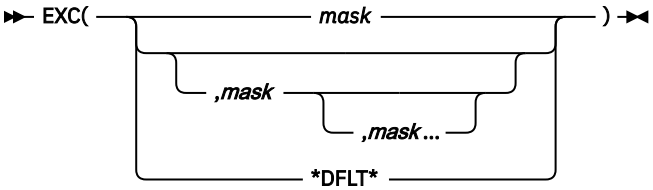
hhmm
Specifies a time duration in hour and minute notation. The value must be four decimal digits. The minimum value is 0001 and the maximum value is 2400. The last two digits (mm) must be in the 00 - 59 range.

The following example code instructs the Usage Monitor to stop after 150 minutes.

```
F HSIJMON,DUR(0230)
```

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.



mask
Specifies a 1 - 8 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 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.

Table 15. Examples of using the EXC command

Command purpose	Example code
Reset the exclusion table to its default status.	`F HSIJMON,EXC(*DFLT*)`
Exclude the collection of data for Language Environment modules and REXX modules.	`F HSIJMON,EXC(CEE*,IRX*)`
Exclude the collection of data for the program CEE.	`F HSIJMON,EXC(CEE)`

HOF - Adjust for hypervisor STCK TOD clock offset

HOF is used to control whether the TOD clock offset in a logical partition is to be applied to collected data, or not. When HOF(N) is set, data timestamps are derived from the local time as supplied by z/OS. When HOF(Y) is set, the hypervisor STCK date and time offset from field SMF89HOF in SMF type 89 records is subtracted from z/OS local time to form the collected timestamp values.

For Usage Monitor data, the HOF setting at the time that the writer task is attached after the closure (or switch) of a collection repository is used.

For Inquisitor data, the HOF setting active at the time of the Inquisitor program initialization is used.

HOF(Y) will not cause any change to data timestamp values unless the Usage Monitor has processed a type 89 SMF record. For this to occur, SMF parameter settings must specify the collection of type 89 records, and at least one SMF interval must have ended while the Usage Monitor is active before the output file data generation commenced.

If the Usage Monitor has been stopped before an Inquisitor scan commences, the Inquisitor program uses the HOF status current at the time of Usage Monitor termination.

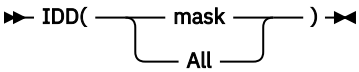
Y
Specifies the hypervisor STCK TOD clock offset will be used to adjust date and time values present in collected data.

N
Specifies the date and time values present in collected data will be based wholly on the local time, as maintained by z/OS.

HOF(N) is the default setting that will be used if no HOF command has been issued.

IDD - Deleting data set name inclusion entries

IDD is used to remove data set name masks previously added by the IDS command. If a mask of *ALL* is specified then all data set inclusion masks are deactivated.



mask
Specifies a 1 - 44 character data set name mask. Any wildcard characters in the mask are treated as literals for the purposes of finding the mask to delete.

The following example code deactivates the SYS3.LINKLIB inclusion mask.

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

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 only affect data collection if there are active user-specified exclusion masks. An inclusion mask is normally expected to match a subset of data set names that would match an exclusion mask.

mask
Specifies a 1 - 44 character data set name mask. Generic wildcard matching allows a percent sign to match any single character, and an asterisk to match any group of zero or more characters. If the mask contains a slash character (/), the value is processed as a UNIX path name mask rather than a data set name mask.

You can use the following example code if your 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.

```
XDS(SYS3.*)
IDS(SYS3.LINKLIB)
```

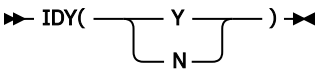
IDY - Collect names created by IDENTIFY

IDY is used to control whether the Usage Monitor will treat entry point names created by the IDENTIFY macro as normal program alias names or replace them with real program names.

Names created by the IDENTIFY service need not match any name that can be found in any program library directory. Each entry point name created by IDENTIFY appears to be an alias of a real program already resident in storage.

From the software asset management point of view, used program names should match the names present in program library directories so that software usage can be assigned to software inventory. When IDY(N) is set, the Usage Monitor will replace names created by IDENTIFY with the name of the program for which the created name is an alias.

From the application functional tracking point of view, it may be preferable to have the names of entry points actually used by the application reported as being used. When IDY(Y) is set, the Usage Monitor will report names created by IDENTIFY as normal alias names. Usage records for aliases also report the real program name in a separate field.



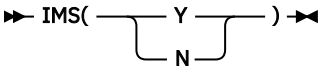
Y
Specifies that entry names created by IDENTIFY will be preserved in collected usage data.

N
Specifies that entry names created by IDENTIFY will be ignored or replaced by the real program name, depending on the event type.

IDY(N) is the default setting.

IMS - Set IMS-managed program usage collection status

Many programs in IMS address spaces are managed by IMS itself rather than by the operating system's program management component. The IMS command provides a system-wide setting to enable or disable the collection of usage of these IMS-managed programs.



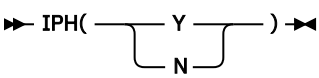
Y
Specifies that IMS-managed program usage is to be collected.

N
Specifies that IMS-managed program usage is not to be collected.

If no IMS command is issued, then IMS-managed program usage is collected. IMS(Y) is the default setting.

IPH - Control collection of TCPIP Host details

JIPH is used to control the reporting of the TCPIP host name and IP address. Program usage data sets created by the Usage Monitor will normally have the TCPIP host details present in the header information, but this data can be suppressed by specifying IPH(N).



Y
Specifies that the Usage Monitor will call HSIPHOST to procure TCPIP host details.

N
Specifies that the Usage Monitor will not report TCPIP host details in the program usage files.

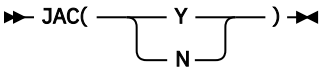
If no IPH command is issued, the default is IPH(Y).

JAC - Set job account collection status

JAC is 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. This 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 can significantly increase data volumes.

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



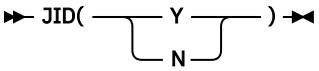
Y
Specifies that job account codes are used.

N
Specifies that job account codes are ignored.

If no JAC command is issued, then job accounts are used. The default is JAC(N).

JID - Control the preservation of batch job identifiers

JID is used to control whether all batch job identifiers are to be preserved or not. Normally usage data for each program is aggregated by job name and user ID, with only the most recent job identifier being retained. JID provides the option of keeping all batch job identifiers so that the number of jobs using a program can be counted, and usage can be attributed to specific individual jobs. Job identifier aggregation for started tasks and TSO user sessions is always equivalent to JID(N) and is not affected by this setting.



Y
Specifies that batch job identifiers should not be overlaid and that different batch job identifiers should prevent data aggregation.

N
Specifies that normal aggregation by job name and user ID is to proceed without considering job identifier differences.

The default setting of JID(N) applies each time the Usage Monitor is started.

JNM - Control the collection of job names

JNM is used to specify whether the Usage Monitor collects the names of jobs which use programs or not. If the names of jobs which use the various programs are not considered to be important, you can dispense with the collection of these names. The advantage of not collecting individual job names is the reduction in processing times and data volumes caused by the aggregation of data into fewer records. When individual job names are not collected, usage is summed over broad address space categories, such as JOB, STC, TSO, and SYS. The total usage counts collected by the Usage Monitor for each program are not affected by this setting.

A change to this setting takes effect at the next collection repository switch.

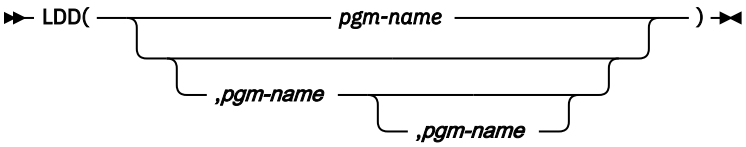
Y
Specifies that the name of each job running a program is to be collected.

N
Specifies that only a broad address space category of each job running a program is to be collected, instead of the individual job name.

If no JNM command has been issued, then job names are collected. JNM(Y) is the default.

LDD - Deactivate LOAD exclusion entries

LDD is used to deactivate a LOAD exclusion entry previously activated by the LDX command. Names specified as operand values should exactly match names already used in LDX commands.



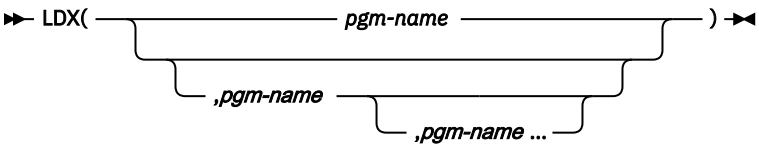
pgm-name
Specifies a 1 - 8 character program name.

LDX - Add or activate a LOAD exclusion

The Usage Monitor tracks programs that are given control as well as those that are loaded into storage even if they are not explicitly given control. When these events occur, the programs involved are deemed to be ***used***. However, there are some auditing and administration software components which bring modules into storage for the purposes of analysis or extracting product enablement status information. In these cases, it is not helpful to say that the target programs were used.

The LDX command is used to exclude these cases from collected usage data. The relevant program names are supplied as values in LDX commands. The programs named in LDX commands are those programs which perform the analysis or which issue the status request of numerous and perhaps varying subject programs.

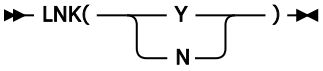
Programs which are the subject of LINK or LOAD and DELETE requests issued by a program named in an active LDX setting are excluded from usage data collection.



pgm-name
Specifies a 1 - 8 character program name.

LNK - Set linklist library collection status

LNK is used to specify whether output data sets written by the writer task will include a list of linklist libraries or not. While program HSIPINQ will collect linklist library information, it can only report the libraries for the system on which it executes, and it may not always be run on every system where the Usage Monitor is running. Enabling this data collection allows the data base to reflect the correct linklist status of libraries for each system where the Usage Monitor is active.



Y
Specifies that linklist library information is to be collected.

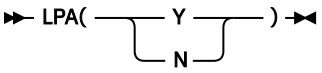
N
Specifies that linklist library information is not to be collected.

If no LNK command is issued, then linklist library information is collected. LNK(Y) is the default setting.

The LNK setting is only referenced by the writer task at the end of a collection cycle, and the linklist library information collected represents the status at that time.

LPA - Set link pack area program monitoring status

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



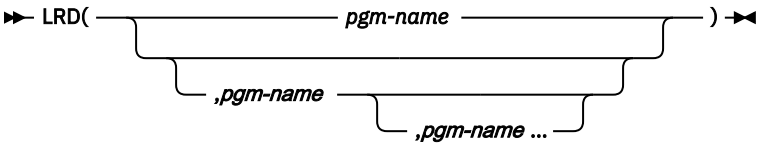
Y
Specifies that LPA program usage is to be monitored.

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

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

LRD - Deactivate a long-running program entry

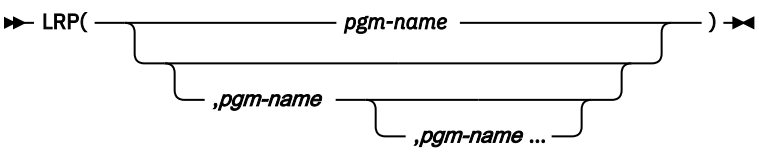
LRD is used to deactivate an entry created by a previous LRP command, or an entry created from the system program properties table. Job step programs with names matching inactive entries will not be treated as long-running programs by the Usage Monitor. Names specified as operand values should exactly match names specified either in a previous LRP command or in the system's program properties table.



pgm-name
Specifies a 1 - 8 character program name.

LRP – Specify long-running program

LRP is used to inform the Usage Monitor that the named program is to be considered as long-running and that the CPU time for the address space is to be entirely attributed to the named program. While this does mean that CPU time for dynamically invoked programs will not be tracked, it does mean that CPU consumption can be monitored without having to wait until the job step terminates. You can use the LRP(*DFLT*) command to undo any LRD and LRP commands that have been processed.



pgm-name
Specifies a 1 - 8 character program name.

The initial list of long-running program names is formed from the system's program properties table active at the time that the Usage Monitor is started.

MCL – Set the data set SMS management class

The MCL command is used to override the management class that is assigned to dynamically created output data sets by the system's ACS routines. You may use this, for example, to inhibit backing up the data sets, or to assign a relevant space management policy.

MCL(*mgmtclas*)

mgmtclas
Specifies a 1 - 8 SMS management class name.

PAK – Set the zip compaction level

The PAK command is used to set the level of zip compaction that the writer task will use when writing to the dynamically created output data sets. The level is specified as a single decimal digit in the 0 to 9 range. Level 0 specifies that the zip ***shrink*** method is used. Levels 1 to 9 specify that the corresponding compaction level of the ***deflate*** method is used. The default is 1 which is the fastest ***deflate*** compaction level.

Larger PAK values will increase data compression but will also consume disproportionately more CPU time.

PAK(n)

n
specifies a single decimal digit from 0 to 9.

If no PAK command is issued then PAK(1) will be used, resulting in a fast ***deflate*** zip file.

PLN - Set the sysplex name

PLN is used to override the name of the sysplex contained in the output header record. The actual sysplex name is used as a norm, but an override allows control over which systems have their inventory updated when the PLX=Y Inquisitor setting is used. The value specified here should match the PLEXNAME= value specified for the corresponding Inquisitor scans. Overriding the sysplex name is not usually needed unless PLX=Y is used and the sysplex grouping does not match the shared DASD grouping. Symbols can be employed in the construction of the sysplex name. Available symbols include all z/OS system symbols, &SMF, the SMF identifier for the system, and &SYSLPAR, the logical partition name for the system.

PLN(plexname)

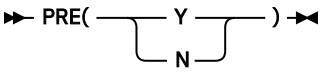
plexname
Specifies a string which is to be resolved to an identifier 1-8 bytes in length.

PRE - Collect usage for long running programs

PRE is used to specify if the Usage Monitor is to collect usage for programs which started before the current collection cycle. Without this data collection a Usage Monitor collection cycle will have no usage data for programs which started running before the cycle started and remain running when the cycle ends. If a job or task runs for more than two days, most days will not have any usage recorded for the main program unless this additional data collection is enabled.

When the additional data collection is enabled, previously fetched programs resident in the regions of started task and batch job address spaces where SMF interval recording is active have usage recorded in each collection cycle which encompassed the end of at least one SMF interval.

This setting can affect usage figures. For example, the main program of a constantly running task can accrue a usage count of around 30 over a month even though it was really only used once for an extended period.



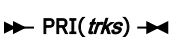
Y
Specifies that usage for previously running programs is to be collected.

N
Specifies that usage for previously running programs is not to be collected.

The default setting of PRE (Y) applies each time the Usage Monitor is started.

PRI - Set the data set space primary allocation

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



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

If no PRI command is issued, the primary space allocation is 750 tracks. The Usage Monitor uses the RLSE space allocation attribute.

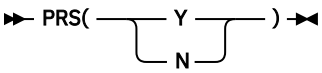
The following example code sets the primary space allocation to 900 tracks.

```
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 the activity of registered software. Registered software uses the system Register service. The data contains information about the usage of registered software, and information about software registration settings from the PARMLIB member IFAPRDxx.

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



Y
Specifies that registered software information is collected and output.

N
Specifies that registered software information is neither collected or output.

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

QSZ - Specify collection element queue area size

QSZ is used to specify the virtual storage size of the SCOPE=COMMON memory object which forms the area where collected usage data is queued to the Usage Monitor address space for storing into the collection repository. The QSZ value specifies the number of storage segments the area occupies, where a segment is one megabyte in size.

The QSZ value used is fixed for the life of the Usage Monitor address space. To change the QSZ value the Usage Monitor started task must be recycled.

►► QSZ(*segments*) ►◄

segments
Specifies a number of segments from 1 to 200.

If no QSZ command is issued, a 10MB queue area will be used. The queue area is processed as a LIFO stack, which means that only the necessary number of pages needed to hold the peak queue length will need to be backed by physical storage, no matter how large the QSZ value is set.

REF - Refresh Usage Monitor settings

REF is used at any time to reset Usage Monitor settings according to commands in the UMONIN file, without stopping and starting the Usage Monitor. The detailed results of the refresh operation are written to the UMONMSG file.

The processes of a refresh operation include:

- Verify that UMONIN is still allocated.
- Open UMONIN.
- 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 UMONIN.
- Close UMONIN.
- Issue either "HSIZ059I" on page 286 or "HSIZ060I" on page 286, as appropriate.

►► REF ►◄

The following example code changes Usage Monitor settings to updated values from UMONIN.

```
F HSIJMON,REF
```

SCL – Set the data set SMS storage class

The SCL command is used to override the storage class that is assigned to dynamically created output data sets by the system's ACS routines.

►► SCL(*storclas*) ►◄

storclas
specifies a 1 – 8 character SMS storage class name.

SEC - Set the data set space secondary allocation

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

►► 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. The Usage Monitor uses the RLSE space allocation attribute.

The following example code sets 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 enables 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 system symbols, &SMF, the SMF identifier for the system, and &SYSLPAR, the logical partition name for the system.

►► SID(*sid*) ►◄

sid
Specifies a string which is to be resolved to an identifier 1-4 bytes in length.

Table 16. Examples of using the SID command

Command purpose	Example code
Set the output system identifier to PROD.	`F HSIJMON,SID(PROD)`
Set the header record system identifier to the current LPAR name. The LPAR name must not exceed four characters in length.	`F HSIJMON,SID(&SYSLPAR)`

SIZ - Set the data space repository size

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

►► 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 232 bytes and contains a pointer to a separate part of the repository dedicated to holding data set and UNIX file names. Storage is conserved by only storing a single copy of each collected data set and UNIX file name. As each repository page has data placed in it for the first time, that page must be backed physically by the system. When a collection repository is full, a repository switch is triggered automatically. A repository switch also occurs when data stamped after the switch time is detected, or when a manual switch is requested by the **SWI** command.

The following example code sets the size of future collection repositories to 1,000,000 entries.

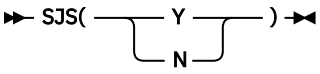
```
F HSIJMON,SIZ(1000000)
```

SJS - Controlling spawned job suffix preservation

When a spawned address space is created by a unit of work with a job name that is shorter than eight characters, the system appends a sequence digit in the 1 to 9 range to the job name, and this becomes the job name of the spawned address space. This approach means that the usage of programs generated by jobs with a specific name can be logged under as many as ten different job names. The system-generated job names usually do not assist in identifying the source of the work because there is often no other reconciliation data which also uses these generated names.

The SJS setting can be used to remove the spawned sequence number suffix so that all usage events for programs are logged under the original job name, resulting in fewer Usage Monitor records and reduced

processing time. If the spawning job name is eight characters long and ends in a digit in the 1 to 9 range, then activity in spawned address spaces (but not the original address space) can be reported under a job name which is only the first seven characters of the original job name. If this is likely to present a problem then use SJS(N).



Y
Spawned job name suffix digit is truncated.

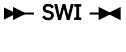
N
No editing is performed of spawned address space job names.

SJS(Y) is the default if no SJS command has been issued since the Usage Monitor started.

SWI - Switch to a new collection repository

SWI causes a new collection repository to be created and used for subsequent data collection. A writer task processes the data contents of the repository that is being used at the time that the SWI command is issued.

The SWI command has no operands. It is invalid in the UMONIN 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. The SWI command might be rejected if the writer task is busy.

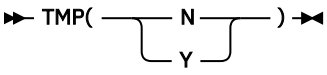


The following example code manually switches to a new repository.

```
F HSIJMON,SWI
```

TMP - Set temporary data set collection status

TMP is used to specify how program libraries with system generated names are to be processed. Normally the Usage Monitor discards information about programs fetched from temporary data sets since the library will no longer exist after the job ends and so will not form part of the persisting software inventory. However, if you wish to collect usage data from programs residing in temporary data sets then you can set TMP(Y).



Y
Specifies that usage containing temporary data set names is collected.

N
Specifies that usage containing temporary data set names is discarded.

If no TMP command is issued, then program usage events pertaining to temporary data sets are discarded. TMP(N) is the default setting.

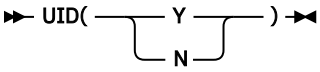
UID - Control the collection of user details

UID is used to specify whether the Usage Monitor collects the identifiers and names of users who use programs or not. If the details of users who use the various programs are not considered to be important, then you can dispense with the collection of this information. The advantage of not collecting user information is the reduction in processing times and data volumes.

When user information is not collected, the user ID data item remains blank, and user names are not output, regardless which UNM setting is current. The total usage counts collected by the Usage Monitor for each program are not affected by this setting.

If you want program usage attributed to individual users but do not want the names of users to be retained, use UID(Y) and UNM(N).

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



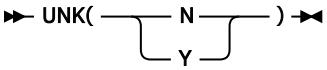
Y
Specifies that details of each user using a program are to be collected.

N
Specifies that details of each user using a program are not to be collected.

If no UID command is issued, user details are collected. UID(Y) is the default.

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. It must be set only when requested by IBM support.



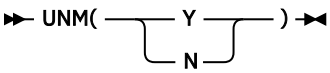
Y
Specifies that the "unknown" events are to be collected.

N
Specifies that the "unknown" events are not to be collected.

If no UNK command is issued, the unknown events are not collected. UNK(N) is the default setting.

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 eight 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 are output. The output of the user ID is controlled by the **UID** setting. This setting is checked by the writer task when the data in a collection repository is being processed for output.



Y
Specifies that collected user names are written to the output file.

N
Specifies that collected user names are discarded.

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

UNT - Set the data set allocation unit

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

unitname
Specifies a 1 - 8 character long unit name.

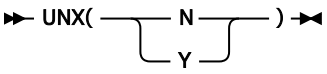
If no UNT command is issued, SYSALLDA is used.

The following example code sets the allocation unit to WORKDA.

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

UNX - Set UNIX program monitoring status

UNX is used to determine if the programs retrieved from Hierarchical File System (HFS) files are to be monitored.



Y
Programs fetched from HFS files are to be monitored.

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

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

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.

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

If no VOL command is issued, a specific volume is not explicitly requested. You must then have PUBLIC or STORAGE volumes in the public allocation pool, unless the data sets are managed by SMS.

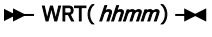
The following example code sets the allocation volume to SCR001.

```
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 collection repository, and automatically switch to a new one. The data write-out for the closed repository is also initiated at the same time. These events are triggered when data from after the specified time is detected.

The UTC or GMT switch time is calculated using the local time when the repository is created. The time that a data space is 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 repository is closed. The time of the switch, after the next switch, is calculated using the new local time.



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

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

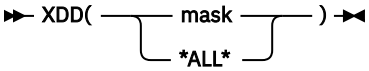
The following example code sets the automatic switch-and-write time to 10 minutes before midnight.

```
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. XDD can also deactivate entries from the default exclusion list that was automatically created by the Usage Monitor.

If a mask of *ALL* is specified then all (both default and user) data set exclusion masks are deactivated.



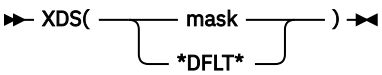
mask
Specifies a 1 - 44 character data set name mask. Any wildcard characters in the mask are treated as literals for the purposes of finding the mask to delete.

The following example code deactivates the SYS3.* exclusion mask.

```
F HSIJMON,XDD(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 user-defined exclusion mask list. Usage data matching active default data set exclusion masks is discarded before inclusion masks created by IDS commands are examined. Any inactive default exclusion masks can be reactivated by the XDS(*DFLT*) command which does not affect user-specified data set filtering.



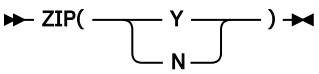
mask
Specifies a 1 - 44 character data set name mask. Generic wildcard matching allows a percent sign to match any single character, and an asterisk to match any group of zero or more characters. If the mask contains a slash character (/), the value is considered to be processed as a UNIX path name mask rather than a data set name mask.

The following example code excludes program usage data from collection for programs fetched from data sets with a high-level qualifier of SYS3.

```
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 volume, in turn reducing data transfer time and storage space requirements.



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. ZIP(Y) is the default setting.

If abend S213-C8 occurs when the writer task opens the output data set, **ZIP(N)** will be issued internally and the data will be written without being zipped. In this case, the following WTO message will be issued:

```
HSIZ033I UMON-WTR ZIP(N) COMMAND PROCESSED
```

Usage Monitor Trace Facility

Trace Facility function

While accumulating program usage data intended for database import, the Usage Monitor performs some basic data aggregation with counters being maintained so that multiple similar events can be represented by a single record with an associated event count.

For those occasions where you want to track individual program management events and ascertain their precise chronology, the Usage Monitor Trace Facility can be used to collect this information, with dynamically alterable filters and without affecting the data collected for the data base.

Trace data flow

Trace data is written to a separate data set in real time so there is no increase in the volume of program usage data staged in virtual storage. It is even possible to trace events which are excluded from Usage Import processing.

Once a trace is active, a record will be written for each event that matches the selection criteria. When no space remains in the output data set, the trace will be terminated without affecting the main data collection. The output data set is deallocated by the Usage Monitor whenever tracing terminates.

Trace data is not affected by a collection repository switch or by writer task activity. Any active trace will be terminated when the Usage Monitor terminates.

Data collected by the Trace Facility

Tracing selection criteria is specified as one or more job name masks, one or more program name masks, or both. Tracing will occur if the job name matches any active job name mask, or if the 8-byte program name matches any active program name mask. Currently there is no trace filtering based on UNIX program names, but UNIX program events can be traced based on job name filtering.

The data collected for Usage Import contains several different record types, but the type with the largest number of records is usually the program usage detail record. It is this record type that is written as trace data. Each record is appended with a suffix which contains additional data not needed by Usage Import, such as the program event type and a STCK event timestamp.

Trace data set contents

Trace data will be written sequentially using QSAM as each record presents. Because all records will have the same structure, fixed-length records are used. The Usage Monitor will override the logical record length to the required value and will reset the block size to zero to ensure that the system-determined block size is used. Although the Usage Monitor will not compress this data by zipping it, the data can be compressed by using system-managed storage facilities where available.

While HSIIBRWZ (shipped in the SHSIEXEC library) is intended to be a general zip file browser (using the ISPF BRIF service), it does also function with unzipped data. The HSIBRZIP program (which HSIIBRWZ calls) will insert a data item label line for Usage Monitor trace records in the display as a browsing aid, as it does with any recognized Usage Monitor or Inquisitor record.

Member HSISTRCD shipped in the SHSISAMP library contains assembler source describing the structure of Usage Monitor trace records.

Usage Monitor trace commands

Commands for the Usage Monitor Trace Facility are listed in this section.

TRCD-A
Display data set allocation parameters.

TRCD-J

Display trace selection job name masks.

TRCD-P

Display trace selection program name masks.

TRCDCL

Set output data set SMS data class.

TRCDSN

Set output data set name.

TRCEXC

Control tracing of excluded events.

TRCJOB

Specify job name trace selection filter.

TRCJOD

Delete job name trace selection filter.

TRCMCL

Set output data set SMS management class.

TRCOFF
Terminate tracing.

TRCON
Initiate tracing.

TRCPGD

Delete program name trace selection filter.

TRCPGM

Specify program name trace selection filter.

TRCPRI

Set output data set primary space in tracks.

TRCSCL

Set output data set SMS storage class.

TRCSEC

Set output data set secondary space in tracks.

TRCSTA

Set output data set initial disposition.

TRCUNIT

Set output data set device allocation unit.

TRCUNX
Control UNIX syscall trace.

TRCVOL

Set output data set volume serial.

The syntax rules described in Usage Monitor general commands also apply to the trace commands.

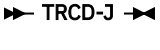
The description of each trace command follows.

TRCD-A - Display data set allocation parameters

TRCD-A is used to display dynamic allocation details to be used in the creation of trace output data sets. The data set name, primary and secondary space quantities, and optional SMS class names, unit and volume serial number are shown.

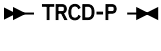
TRCD-J - Display trace selection job name masks

TRCD-J is used to list the active trace job name selection masks. Events for any program in address spaces (except some excluded system address spaces) with names matching any mask in this list will be logged to the trace data set.



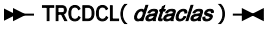
TRCD-P - Display trace selection program name masks

TRCD-P is used to list the active trace program selection masks. Events in any address space (except some excluded system address spaces) for programs with names matching any mask in this list will be logged to the trace data set. Only 8-byte MVS program names (as opposed to UNIX path names) are examined for this selection filtering



TRCDCL - Set output data set SMS data class

The TRCDCL command is used to specify the SMS data class for the trace output data set.



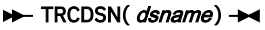
dataclas
specifies a 1-8 character SMS data class name. Omit to clear a previous setting.

TRCDSN - Set output data set name

The TRCDSN command is used to specify the name of the trace output data set.

Symbols can be used to form the data set name used at allocation time. Available symbols are system symbols, &SYSPLAR and &SMF. &SYSLPAR (the current LPA name) may resolve to a null string when z/OS is running under z/VM. &SMF resolves to the SMF identifier of the system.

Unlike the DSN command, the symbols are resolved each time a trace session is initiated, meaning that dynamic system symbols can be used to generate a unique data set name for each trace session.



dsname
specifies a 1-44 character data class name.

If no TRCDSN command is issued, a data set name of NULLFILE will be used.

The following example sets the output data set name to SYS2.UMTRACE:

```
TRCDSN(SYS2.UMTRACE)
```

The following example uses symbols to make the data set name reflect where and when the trace data was collected:

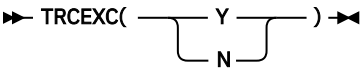
```
TRCDSN(UMONHLQ.Z&SMF..UMTRACE.D&LDATE..T&LTIME.)
```

Note the following points pertaining to the previous example:

- If this command is passed to the Usage Monitor using the system MODIFY command, system symbols will be resolved by the system before the Usage Monitor receives the command string. In this example, &LDATE and &LTIME will each be converted to a 6-numeric character constant, while &SMF will remain unresolved by the system because it is not a system symbol.
- To have system symbols dynamically interpreted at the time the data set is allocated, the TRCDSN command must be passed to the Usage Monitor via the UMONIN DD, either at initialization time, or by using the REF command to refresh settings.
- TRCDSN command processing attempts to resolve the symbols to validate the command, and any success message such as message HSIZ0331 will echo the resolved name, but the name to be used when the TRCON command is issued will be stored with symbols unresolved so that the symbols can be reevaluated at data set allocation time.
- For system levels prior to z/OS 2.5, replace &LDATE with &LYYMMDD and replace &LTIME with &LHHMMSS.

TRCEXC - Control tracing of excluded events

TRCEXC is used to control the tracing of events that are excluded from the normal data collection intended for the database.



Y
Specifies that excluded events are to be traced.

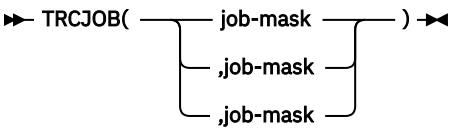
N
Specifies that excluded events are to not be traced.

If no TRCEXC command is issued, then excluded events are not traced. TRCEXC(N) is the default setting.

When considering the use of TRCEXC(Y), be aware of the potential increase in overhead due to the processing of additional events that would normally be excluded.

TRCJOB - Specify job name trace selection filter

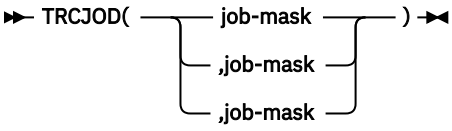
TRCJOB allows additional job name trace selection filters to be activated. Program usage events detected in an address space (apart from various system address spaces) with a name that matches any of the active job name selection masks will be traced.



job-mask
Specifies a 1 – 8-character job name mask. Each mask can be made generic by including one or more generic masking characters. A percent sign (%) may be used in the mask to match any value of the corresponding single job name character. The mask may also end with an asterisk (*), indicating all further characters until the end of the job name are considered to match. Note that a mask of only a single asterisk will match all job names, meaning that all detected events will be traced. Depending on system workloads, tracing all detected events may add noticeable overhead to system and/or application processing.

TRCJOD - Delete job name trace selection filter

TRCJOD is used to deactivate one or more job name trace selection filters that were added by previous by a TRCJOB command. The mask value(s) specified must exactly match an active job name filter. If necessary, use the TRCD-J command to display active job name filters.

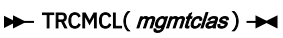


job-mask
Specifies a 1 – 8-character job name mask.

Use the command TRCJOD(*ALL*) to deactivate all trace job name selection filtering.

TRCMCL - Set output data set SMS management class

The TRCMCL command is used to specify the SMS management class for the trace output data set.



mgmtclas
specifies a 1-8 character SMS data class name. Omit to clear a previous setting.

TRCOFF - Terminate tracing

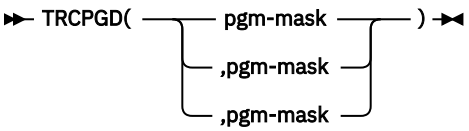
TRCOFF is used to end an active trace, causing the Usage Monitor to close and deallocate the output data set. Issuing TRCOFF when tracing is inactive has no effect.

TRCON - Initiate tracing

TRCON is used to initiate an active trace, causing the Usage Monitor to attach a trace subtask which will allocate the output data set using the current value of relevant settings, and open the data set for output. When the subtask indicates that the data set is ready to accept data, tracing will commence. Issuing TRCON when tracing is already active has no effect.

TRCPGD - Delete job name trace selection filter

TRCPGD is used to deactivate one or more program name trace selection filters that were added by previous by a TRCPGM command. The mask value(s) specified must exactly match an active program name filter. If necessary, use the TRCD-P command to display active program name filters.

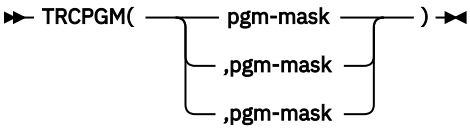


job-mask
Specifies a 1 – 8-character program name mask.

Use the command TRCPGD(*ALL*) to deactivate all trace program name selection filtering.

TRCPGM - Specify program name trace selection filter

TRCPGM allows additional program name trace selection filters to be activated. Program usage events detected for programs with a name that matches any of the active program name selection masks in any address space (apart from various system address spaces) will be traced. Only 8-byte program names (and not UNIX program path names) are examined.



pgm-mask
specifies a 1 – 8-character program name mask. Each mask can be made generic by including one or more generic masking characters. A percent sign (%) may be used in the mask to match any value of the corresponding single program name character. The mask may also end with an asterisk (*), indicating all further characters until the end of the program name are considered to match. Note that a mask of only a single asterisk will match all program names (including UNIX programs), meaning that all detected events will be traced. Depending on system workloads, tracing all detected events may add noticeable overhead to system and/or application processing.

TRCPRI - Set output data set primary space

TRCPRI is used to specify the primary space allocation quantity in tracks for the trace output data set.

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

If no TRCPRI command is issued, a primary space quantity of 750 tracks is used. Unused space is not released when the trace ends so that a practice of retaining a permanent trace data set would be feasible.

TRCSCL - Set output data set SMS storage class

The TRCSCL command is used to specify the SMS storage class for the trace output data set.

storclas
specifies a 1 – 8-character SMS storage class name. Omit to clear a previous setting.

TRCSEC - Set output data set secondary space

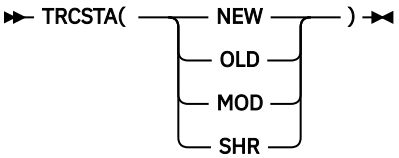
TRCSEC is used to specify the secondary space allocation quantity in tracks for the trace output data set.

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

If no TRCSEC command is issued, a secondary space quantity of 300 tracks is used. Unused space is not released when the trace ends so that a practice of retaining a permanent trace data set would be feasible.

TRCSTA - Set output data set initial disposition

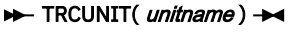
TRCSTA is used to set the initial status of the trace output data set. This is the same data set initial status that is specified by the first subparameter of DISP in a DD statement of JCL.



If no TRCSTA command is issued, a value of NEW will be used. For more information including a discussion on serialization implications, see the section for DISP in z/OS MVS JCL Reference.

TRCUNIT - Set output data set device allocation unit

TRCUNIT is used to specify the unit name to be used when allocating the trace output data set.



unitname
specifies the 1 – 8-character unit name.

If no TRCUNIT command is issued, a unit name of SYSALLDA will be used.

TRCUNX - Control tracing of UNIX syscalls

TRCUNX is used to control the tracing of z/OS UNIX system service calls or syscalls made by applications. For each traced call a trace record is written before the UNIX system service gets control and another record is written when the system service call completes and returns control to the application.

Y
Specifies that UNIX syscalls are to be traced.

N
Specifies that UNIX syscalls are to not be traced.

TRCUNX(N) is the default so if no TRCUNX command is issued, then UNIX syscalls are not traced.

Tracing UNIX syscall events is implemented using the BPX_PRE_SYSCALL and BPX_POST_SYSCALL system exit points. The call types that can be traced are controlled by the list of call stub names present in the EBCDIC text file (conventionally residing in the /etc directory) named in the SC_EXITTABLE setting of an active BPXPRMxx system PARMLIB member. If you change the stub list in the text file, you can get the system to process the update by issuing the SET OMVS=xx system command where xx is the suffix of the relevant BPXPRMxx PARMLIB member.

Trace records generated by the TRCUNX(Y) setting have the following properties:

- Their generation is not affected by the TRCEXC setting.
- They are only generated for callers in task mode.
- They are only generated after a job name mask from a TRCJOB command matches the current address space name.
- They are not affected by program name filtering even though the corresponding stub name is reported in the 8-byte program name field (even when such a stub is not used).
- The first three characters of the 8-byte program name field will be `Pre` for pre-calls and BPX for post-calls.
- The fourth character of the 8-byte program name field will be a 4 for 64-bit callers.
- The provider field has a value of `UNIX` in EBCDIC.
- Unlike other trace records and usage detail records, the CPU time fields (which have a resolution of hundredths of a second) represent the accumulated CPU time of the task at the time the information was collected.
- Unlike other trace records and usage detail records, the program active time field contains the task address rendered in EBCDIC to assist post-processor data collation. (Use the STCK value in the record suffix to perform elapsed time analysis.)
- For those syscalls passed a string such as a user or path name, the string is reported in the file name field. For other syscall types, the file name field contains Callxxxx where xxxx is the hexadecimal call type number. Read and write calls will also have the return value reported after the RV=indicator.
- The volume serial field contains blanks except for those call types where a file descriptor is used. For open (post-call only), close, read, seek, and write call types, the 6-byte volume serial field contains x6FFD followed by the 4-byte file descriptor value to assist post-processor data collation.

TRCVOL - Set output data set volume

TRCVOL is used to direct the allocation of the trace output data set to a specific volume.

volser
specifies the 1 – 6-character volume serial number of the trace data set volume.

If no TRCVOL command is issued then no specific volume is requested. You must then have PUBLIC or STORAGE volumes in the public allocation pool, unless the data set is managed by SMS.

IBM CICS® Transaction Server for z/OS®

The CICS Transaction Server for z/OS performs much of its program management outside of the contents supervisor framework that most applications use. For the Usage Monitor to accurately detect and record the use of programs in a CICS region, you must customize each CICS region where you require detailed program usage monitoring.

To prepare a CICS region to enable detailed monitoring, you must install the following components:

- CICS global user exist (GLUE) programs
- An enabling program to activate these user exit programs
- An entry in the program list table (PLT) that triggers the enabling program

The customized JCLLIB library contains the following sample jobs that you can copy and use in your customization:

- The HSISENAX member contains a sample job to translate, assemble and bind the enabling program.

- The HSISPLTX member contains a sample job to create a PLT with the required entry to trigger the enabling program. If you use this sample job, verify the name of the enabling program and the PLT suffix before you submit the job.

The CICS program monitoring facility does not support releases earlier than CICS Transaction Server 5.1. Different releases of the CICS Transaction Server require different versions of the GLUE programs. You must ensure that the correct version of these programs is used for each CICS region. You must also take care when upgrading regions to a later release of CICS so that the correct version of this module will be used with the newer software.

Installed GLUE modules from older releases of the Usage Monitor will not collect any data for this release of the Usage Monitor. The GLUE modules in this release of the Usage Monitor cannot collect any data for older releases of the Usage Monitor.

The following table lists the required GLUE programs for different versions of the CICS Transaction Server:

Table 17.

CICS Transaction Server release	**Usage Monitor GLUE modules**
5.1, 5.2, and 5.3	HSIZFTC3, HSIZEII3
5.4, 5.5, 5.6, 6.1 and 6.2	HSIZFTC4, HSIZEII4

When you implement this CICS Transaction Server customization, the Usage Monitor can collect and record data related to program name and data set name. The collected data is subject to the Usage Monitor program name and data set name selection and exclusion filters. You can stop data collections from all HSIZFTC*x* and HSIZEII*x* GLUE programs with the CIC(N) Usage Monitor setting. CIC(Y) is the default setting if you do not issue a CIC Usage Monitor command.

Depending upon the level of program usage detail you require, the HSIZFTC*x* exit might produce sufficient data for your needs, without the HSIZEII*x* exit. If you want to access more detailed CICS data, such as particulars of transactions and the end users involved, a specialized CICS monitor such as IBM® OMEGAMON® for CICS on z/OS is required.

Customizing a CICS region to provide usage data

Procedure

1. Copy the appropriate HSIZFTC*x* and HSIZEII*x* global user exit (GLUE) programs from the SHSIMOD1 library to a DFHRPL library of the CICS region.
2. Customize and submit the HSISENAX job to create a program that enables the HSIZFTC*xx* and HSIZFTC*xx* exit programs:
   a) Customize the sample job for translating, assembling, and binding the enabling program that is provided in the HSISENAX member in the customized JCLLIB library.

      For convenience, you can name this program HSIZENA*x*, where *x* is the same suffix character as the suffix of the HSIZFTCx and HSIZEII*x* programs that it enables.
   b) Check that the name specified in the PROGRAM operand of EXEC CICS ENABLE statement is the name of the enabling program.
   c) Check that the name specified in EXEC CICS ENABLE PROGRAM statement is the name of the of the GLUE programs.
   d) Link the HSIZENA*x* enabling program into the same DFHRPL library where you copied the HSIZFTC*x* and HSIZEIIx GLUE programs.
   e) Submit the HSIZENA*x* job.
3. Add an entry in the following format to the active program library table (PLT) of the CICS Transaction Server to install the HSIZENA*x* module:

```
DFHPLT TYPE=ENTRY,PROGRAM=HSIZENA
```

Place the entry before the `DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM` entry to load it early during CICS initialization and minimize the need for program resource definitions.

4. Ensure that the PLTPI setting for the CICS region specifies your newly updated PLT.
5. Optional: Use the HSITAGP tagger program to tag non-vendor application programs that you want to be identified in usage reports.

Results

When you complete this task, the use of programs that are given control by various mechanisms in the CICS Transaction Server are attributed to the CICS region address spaces that invoke them.

What to do next

You can stop data collection from all HSIZFTC*x* and HSIZEII*x* glue programs with the CIC(N) Usage Monitor setting. The CIC(Y) option is the default if you do not issue a CIC Usage Monitor command.

Importing Inquisitor data

The Inquisitor Import reads data from Inquisitor scans, where the data is filtered and matched to products. The filtered, matched data is then copied to the Repository tables where it can be viewed and queried by the Analyzer reporting utility.

An audit table captures information of the IQ Import when this job is run, and these audit records can be browsed using an Analyzer report. As part of housekeeping, obsolete audit records from this program are deleted by the MDEL Deletion utility, together with audit records from other utilities.

Related concepts

PLX parameter of the Inquisitor program
The **PLX** parameter can reduce the time it takes to scan and process different systems that have completely shared DASD, and therefore identical software inventories. When you set **PLX**=**Y**, the Inquisitor Import detects libraries that are shared or libraries that have not changed and quickly processes scans of these shared SIDs.

Related tasks

Collecting scanned libraries with the Inquisitor for z/OS
The Inquisitor is a program that scans and collects information about partitioned data set (PDS) and partitioned data set extended (PDSE) program libraries. The Inquisitor Import program takes the collected data as input to form the basis of your software inventory.

Running the Inquisitor Import

The HSISIQIM job in the JCLLIB library performs the Inquisitor Import. This job is generated from the HSISCUST post-installation customization job.

About this task

Warning: This job will terminate immediately if the GKB version referenced by the job is older than 12 months. To rerun this job you need to:

- Download the PTF containing the latest GKB version.
- Run HSISGKBL to load the latest GKB version.
- Rerun HSISIQIM.

Run-time for the HSISIQIM job depends on the number of modules to be imported into the database Inquisitor tables. Because the processing is memory-intensive, run the HSISIQIM job during off-peak periods.

Procedure

1. In the HSISIQIM job, update the following parameters, according to your requirements:
   - **FULLREMATCH**: Set to Y (default), to process all libraries in the in the repository. For a given system ID (SID), existing libraries in the repository are marked as deleted unless the libraries are found in the scanned Inquisitor file. For shared libraries (where PLX=Y), the scanned Inquisitor file can be from any SID that belongs to the sysplex. For non-shared libraries (where PLX=N), the scanned Inquisitor file must be from the same SID. The deletions include products, libraries, and modules. For more information on the setting of PLX=Y, see Scenario 5: Implementing in a sysplex environment.

     Set to N to skip import from scanned libraries where no member directories have changed since a previous Inquisitor Import to the same Repository.
   - **PRODUCTONLY**: Set to N to import all modules, including unidentified modules. Set to Y to import only matched modules.

   The settings for both of these parameters influence the duration of the import process.
2. Review and modify other parameters, as required, in the PARMLIB member.
3. Db2 only: In PARMLIB member HSISCLI, parameter MVSDEFAULTSSID can be assigned the Group Attach Name for Db2 subsystems configured in a Data Sharing Group. The default value is the Db2 subsystem name.
4. Submit the HSISIQIM job.

Import filters and matching

When you import data collected by the Inquisitor, the import procedure reads the data and filters and matches the data before copying the data to the Repository tables.

The Inquisitor Import loads data and performs the following tasks:

1. Reads Inquisitor data generated from Inquisitor scans. To exclude importing specific libraries, the Inquisitor data is filtered against a set of supplied Inquisitor filter tables. These Inquisitor filter tables are updated monthly, together with the knowledge databases. The filtering excludes, for example, the ISV distribution libraries.
2. Matches load modules to best fitting products at the version, release, and modification (VRM) level. Best matches for modules are found based on module names and sizes, and information in the Global Knowledge Base (GKB) and Local Knowledge Base (LKB). Temporary scorecard tables are used to hold all the possible scorecards for modules in a given library while they are matched.
3. Loads matched load modules, including matching information, into the Repository tables. Data from the Repository tables are now ready for viewing or reporting using the Analyzer reporting facility.
4. Aggregates usage data for rediscovered modules in the Repository tables.

The Inquisitor Import uses memory intensively in order to efficiently match many Knowledge Base scorecards to library modules. The maximum memory requirements depend on the number of modules in a library of an Inquisitor Import file, and the number of scorecards in the GKB and LKB that affect the processed libraries. To estimate a requirement, allow 5M +1.5k per module. For example, for an Inquisitor file containing a maximum library size of 30000 modules, the requirement is approximately 5M + (0.0015*30000) = 50M.

TPARAM parameters

The TPARAM parameters that you specify for Inquisitor Import define what data is included in the import.

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

COUNTUSAGE=

Default setting is COUNTUSAGE=N.

The parameter controls when product usage totals are counted. Setting to Y indicates that records are counted in the Inquisitor Import job for modules that have usage and are identified for the first time when usage records were imported before discovery was performed. These recounts of product usage because of modules that are identified can be time consuming.

The Inquisitor Import deletes count entries for products for which at least one module is identified for the first time, and that have usage details as these counts are invalidated by the identification. This is done regardless of the setting of the COUNTUSAGE parameter.

For performance reasons, if you are running Inquisitor Imports for many systems importing IQ data into the same Repository, run the Aggregator step only once in the last IQ Import job. The Aggregator step performs the same function in summarizing the product usage records.

DSN=
Db2 location. Value assigned, as defined in job HSISCUST

FILTERSCHEMA=
Inquisitor Import filter qualifier. Name of qualifier is *&GKBZSCHM*_IQF7

FULLREMATCH=
Default is Y, which means all libraries are imported and processed. N means to skip import from scanned libraries where no member directories have changed since a previous Inquisitor Import to the same Repository.

GKBSCHEMA=
Global Knowledge Base qualifier for z/OS. Name of qualifier is *&GKBZSCHM*_GKB7

GKUSCHEMA=
Global Knowledge Base qualifier for z/OS UNIX. Name of qualifier is *&GKBZSCHM*_GKU7

LKBSCHEMA=
Local Knowledge Base qualifier for z/OS. Name of qualifier is *&REPZSCHM*_LKB7

LKUSCHEMA=
Local Knowledge Base qualifier for z/OS UNIX. Name of qualifier is *&REPZSCHM*_LKU7

PRODUCTONLY=
Default is N, which means all modules, including unidentified modules, are loaded into the Repository. Y means only modules that have been matched to known products are loaded into the Repository, meaning, application modules are excluded.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

PROCESSSUITES=

Enable/disable product suite processing. Default is 'Y'.

Importing Usage data

The Usage Import job imports data generated by the Usage Monitor and aggregates usage data for discovered or identified modules in the Repository tables in the database.

An audit table captures information of the Usage Import when this job is run, and these audit records can be browsed using an Analyzer report. As part of housekeeping, obsolete audit records from this program are deleted by the MDEL Deletion utility, together with audit records from other utilities.

Related tasks

Collecting usage data with the Usage Monitor
The Usage Monitor is a server address space that runs as a started task. Work is queued to the Usage Monitor from all address spaces where programs are used. The Usage Monitor moves captured program usage data into the collection repository and periodically writes the accumulated data to a sequential file. The Usage Monitor runs APF authorized and is non-swappable.

Running the Usage Import

About this task

The HSISUIMP job in the JCLLIB library performs the Usage import. This job is generated from the HSISCUST post-installation customization job. Because run-time for this job depends on the volume of usage data to load, run this job during off-peak periods.

Usage data files can be either outputs from the Usage Monitor or condensed outputs from the ZCAT utility. These output files can be concatenated as a single input in the job. When you are priming a new repository, use a single small file as input when loading usage data for the first time.

If the HSISUIMP job processes the same data that is generated by the Usage Monitor for a specified system, the job skips with a warning message and continues processing the remaining files.

When running Usage import for an LPAR that belongs to a shared sysplex environment, all library records already identified in the LPAR where the IQ Import was run, are also populated into the LPAR where Usage import was run. For example, if IQ import is run only for MVSA (with PLX=Y), and Usage import for MVSB is run for the first time, then all library records identified in MVSA are now propagated for MVSB. This means that when viewing the Analyzer product inventory report for LPAR MVSB, all products belonging to the shared sysplex are displayed. This propagation of library records does not occur if PLX=N is set in MVSA.

For Db2 only: If you are running Db2 in a Data Sharing Group, you can use the Group Attach Name. See Step 2 of Importing Inquisitor data.

TPARAM parameters

The TPARAM parameters that you specify for Usage Import define what data is included in the import.

CHECKPERIOD=
Default is 12. Highlights records from the TUSEMTD table with periods that are older than the CHECKPERIOD value (months). For performance reasons, run the Usage deletion job to delete obsolete Usage records; that is, records that are older than the CHECKPERIOD value.

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

COUNTUSAGE=
Default setting is COUNTUSAGE=N.

The parameter controls when product usage totals are counted. Setting to Y indicates that records are counted in the Usage Import job for systems and periods that are processed and receive new usage details.

The Usage Import deletes count entries for systems and periods that receive new usage details and are invalidated by the new details, regardless of the setting of the COUNTUSAGE parameter.

For performance reasons, if you are running Usage Imports for many systems importing usage data into the same Repository, run the Aggregator step only once in the last Usage Import job. The Aggregator step performs the same function in summarizing the product usage records.

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

GKBSCHEMA=
Global Knowledge Base qualifier for z/OS®. Name of qualifier is *&GKBZSCHM_GKB7.*

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

Aggregating usage and discovery data

The Aggregator is a program that populates the Asset tables.

An audit table captures information of the Aggregator when this program is run, and these audit records can be browsed using an Analyzer report. As part of housekeeping, obsolete audit records from this program are deleted by the MDEL Deletion utility, together with audit records from other utilities.

These tables are accessed by Analyzer reports and batch jobs, and offer a higher level view of product discovery and usage at a version rather than a release or module level. The Aggregator also calculates product discovery and usage count totals that are used to speed up the Analyzer queries.

The Aggregator should be run as the final job in a batch run of Inquisitor Import and Usage Import jobs, and before the repository is accessed by the Analyzer. This ensures that asset reports and counts are synchronized with the latest collected discovery and usage details.

It is usual for asset level usage details to be kept longer than module usage details. This is controlled by the KEEPAGGR parameter of the Usage Deletion program, which has a longer default period than the KEEPDETAIL (or FIRSTDATE/LASTDATE) parameter. The aggregated entries are normally kept for the longer KEEPAGGR period and are usually only recalculated for periods where module usage details are available. Note that the KEEPAGGR parameter value determines the number of periods of Asset (aggregated entries) records to be retained when the COUNTUSAGEFULL=Y is set. See description of parameter COUNTUSAGEFULL in the TPARAM section.

The Aggregator run can be time consuming since it is run for the entire repository and periods rather than an individual SID (as in the case of the Inquisitor Import), or system and period (as in the case of the Usage Import). It is therefore more efficient to run the Aggregator only once following several imports of usage or discovery data. Ensure also that you do not keep usage details for longer than is necessary for your site, and that Usage Deletion is run on a regular basis.

An example of how to run the Aggregator can be seen in sample jobs that are generated by HSISCUST, such as HSISIQIM (Inquisitor Import) and HSISUIMP (Usage Import).

Note Db2 only:

The Aggregator program uses declared global temporary tablespace (DGTT) to process and summarize data in the repository database. And there are also extensive data sorts involved. For performance reasons, increase the sizes of the tablespaces belonging to the work file database and, also consider using different definitions for the Bufferpool (instead of the default BP0 or BP32K). In PARMLIB member HSISAGP1, you can change the storage group and index Bufferpool values for the DGTT (instead of the default BP0 or BP32K).

TPARAM parameters

The TPARAM parameters that you specify for aggregating usage and discovery data is included in the import.

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

COUNTUSAGEFULL=
Default value is N, and this means that only entries for systems and periods that are affected by recent activities are calculated. This parameter performs the same function as the COUNTUSAGE parameter, as described in sections - Importing Inquisitor data, and Importing usage data.

A value of Y means that usage count totals for all systems and periods are deleted and recalculated.

Data are more reliable and up-to-date if the Aggregator is run with a setting of COUNTUSAGEFULL=Y. However, this process is time-consuming. The usage counts can be seen in the Aggregator log.

When COUNTUSAGEFULL is set to Y, the records in tables, TUSEPOV, TUSEPOVLIB, PRODUCT_USE, and PRODUCT_USE_DETAIL, are refreshed, but in different ways. The two summarized tables, TUSEPOV and TUSEPOVLIB are refreshed (first deleted and then recalculated) for all periods that are found in the TUSEMTD table. This means that if there are 3 periods in the TUSEMTD table, then

only data from the 3 periods (and the current month) of information are calculated and stored in the summarized tables. However, for the two Asset tables, PRODUCT_USE and PRODUCT_USE_DETAIL, the refresh process is somewhat different. If there are 12 periods (based on the KEEPAGGR setting in the Usage Deletion job) in these two Asset tables, and TUSEMTD has only the 3 periods, then only data from the 3 periods (and the current month) in the Asset tables are refreshed. Data from the other 9 months are not refreshed (unchanged).

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

IXBUFFERPOOL=
The buffer pool for indexes created on declared global temporary tables that are used by the Aggregator. If not supplied, the buffer pool defined for the database where the global temporary tables belong is used. This value is normally only needed by sites which have a requirement to specify different buffer pool values for different applications. For example, set IXBUFFERPOOL=BP1, instead of the default BP0.

IXSTOGROUP=
The storage group for indexes created on declared global temporary tables that are used the Aggregator. Default is SGHSIIDX. This value need only be specified by sites that have a procedural requirement of not using the default Db2 storage group value, and have used a different STOGROUP value during the creation of the product database repository.

GKBSCHEMA=
Global Knowledge Base qualifier for z/OS. Name of qualifier is *&GKBZSCHM*_GKB7.

GKUSCHEMA=

Global Knowledge Base qualifier for z/OS® UNIX™. Name of qualifier is *&GKBZSCHM_GKU7*.

REPSCHEMA=

Repository qualifier. Name of qualifier is *&REPZSCHM*.

Activating the Automation Server

The Automation Server discovers new data sets and processes them by starting a set of predefined actions that associate the data with data set name masks that form a catalog search. This search determines if any data sets with names matching the mask are to be processed.

Automation Server overview

The Automation Server provides the ability to select data sets if you have data set names that are variable, such as those created by the Usage Monitor, which have low-level qualifiers containing time stamps.

The Automation Server runs as a started task in its own address space.

The user ID for the Automation Server must have an OMVS segment and a UID, or there must be a default UID configured.

The Automation Server issues a system-wide ENQ under the queue name of HSIZNQAS to ensure that there is only one instance of it in a z/OS image. A single instance of the Automation Server continuously references all data sets, catalogs, and volumes that are accessible from all systems in a sysplex so it is unnecessary for the Automation Server to run on more than one system.

Usually a single instance of the Automation Server is sufficient to handle the work from multiple systems which share the same DASD. JOB actions can have any necessary system affinities coded in their JCL.

Input control statements define the processing to be performed by the Automation Server. There are two types of control statements, *action* statements and *DSN* statements:

action
Action statements name the template member which forms the basic input for the action to be performed when a relevant data set is newly discovered by a catalog search. They have optional operands to specify time-of-day, day-of-week, day-of-month, and month-of-year restrictions.

DSN
DSN statements provide a data set name mask to be associated with the preceding action statement. There can be many DSN statements after each action statement.

There are currently two types of action statement:

FTP
Starts the FTP utility to perform a file transfer.

For the FTP action, the template member is read and, after symbol substitution processing, is written to the file defined by the INPUT DD statement. The INPUT file is allocated to a temporary data set. The FTP program is attached as a subtask and scans the INPUT file to process the FTP requests. The report messages it generates are written to the OUTPUT file.

Upon completion of the FTP subtask, the Automation Server examines the completion code. If the program ends normally with a zero return code, the Automation Server deems the action to have been successful and updates the action status in the ASCDS file so the action is not repeated for this data set.

If the FTP program abends, the Automation Server deems the action to have failed. A failed transfer is tried again at a later time. A retry is subject to specified scheduling constraints. The OUTPUT FTP report file contains information to track the exact cause of a transfer failure.

JOB
Submits a batch job.

For the JOB action, the template member is read and, after symbol substitution processing, is written to the file defined by the INTRDR DD statement. This file is directed to the internal reader used by the system, and the jobs submitted by the Automation Server become available for JCL conversion as soon as the INTRDR file is closed, or another JOB card image is found by the reader.

The Automation Server deems all JOB submissions successful, so there are no retries unless the job itself executes the Automation Server Utility program to self-report an unsuccessful run. Any failure should be investigated using the appropriate procedures used by your installation.

Note: The job stream in a JOB action template member may define more than one job.

The Automation Server does not check template member records for either FTP or JCL validity.

Before initiating an action for a detected data set, the Automation Server attempts to exclusively allocate the data set. If the data set is in use, the action is not performed and the awaiting retry status is stored in the control data set. The action is performed when the data set is next found to be available for use within any specified scheduling restrictions. Dynamic allocation failures due to other reasons do not inhibit the action.

Running the Automation Server

To run the Automation Server, you must create a control data set, configure the Automation Server as a started task, design request control statements, and exclude data sets from processing.

Creating the Automation Server control data set

To create the Automation Server control data set, use member HSIASALC in the JCLLIB. This member is generated from the HSISCUST post-installation customization job.

Procedure

1. Allocate sufficient space for the Automation Server 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.

2. Create the control data set by running the HSIASALC job that contains the IDCAMS JCL and control statements.
3. Using the ASCDS ddname, allocate the VSAM KSDS control data set to the Automation Server.

Copying the started JCL task to a library

The Automation Server is run by the HSIJAUTO member in the JCLLIB library that is supplied in the SHSIPROC data set. To start the HSIJAUTO member as a started task, you must customize the JCLLIB member and copy it to an authorized PROCLIB library.

Procedure

1. Review the HSIJAUTO member in your customized JCLLIB library to ensure that the JCL is suitable for your operational environment, making alterations as necessary. If sourcing the JCL directly from the SHSIPROC library, replace the symbols with appropriate values.
2. Copy the HSIJAUTO member from the JCLLIB library to an authorized PROCLIB library.

Files used by the Automation Server

Several files must be available for use by the Automation Server.

STEPLIB
: Load library containing the product software. Not required if IBM Z Software Asset Management is installed into the system link list.

PARMLIB
: Partitioned data set containing fixed-length 80-byte records. Member HSIAPARM of this partitioned data set contains the Automation Server control statements that specify the actions to be performed. For each action in the HSIAPARM member, there is a corresponding member of the same name containing the template data for that action. The template data is made up of JCL or an FTP command stream containing symbolic references, to be resolved by the Automation Server when the action is performed.

ASCDS
: A VSAM KSDS control data set used by the Automation Server.

SYSPRINT
: Specifies the message report file for the Automation Server. Initialization statements, error messages, and activity logging messages, are written to this file.

SYSOUT
: Specifies the message report file for Language Environment.

OUTPUT
: Specifies the message report file for the FTP program. The 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 the Automation Server FTP action is performed.

INTRDR
: Specifies a fixed length 80-byte record file to be directed to the internal reader used by the system. The Automation Server writes a job stream to this file whenever a JOB action is to be performed.

Designing request control statements

Automation Server action requests are specified in the HSIAPARM member of the PARMLIB file.

Syntax rules

Syntax rules are as follows:

- Records with an asterisk in column 1 are comments.
- Blank records are comments.
- A parameter record has one or more parameters, each with a value specified within parentheses after the parameter name.
- The first parameter specifies the statement type.
- All parameters must begin before column 72.
- Blanks can be used before and after parameter names, parentheses, and parameter values.
- Continuations on to subsequent records are not possible.

Statement syntax

Action statement

Each statement requests that an action is performed for a data set when it matches an associated data set name mask, and is detected for the first time. An action is performed once for each match, but the presence of a data set triggers the action for each specified data set name mask it matches.

Action statements have several optional operands to provide control over when Automation Server processing is to occur.

These operands can specify:

- time-of-day window
- day-of-week control string
- day-of-month window
- month-of-year control string

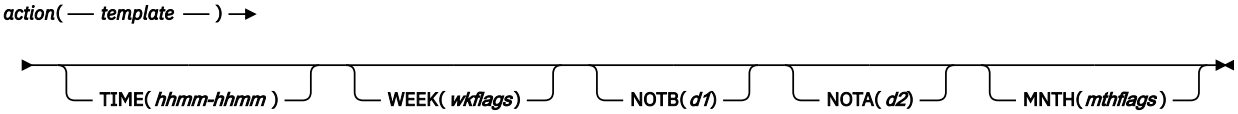
When all these constraints have been satisfied, the Automation Server searches the catalog for data sets with names that match the masks associated with an action.

Data set name mask statement

Each data set name mask statement associates the specified data set name mask with the preceding action statement. Multiple consecutive DSN statements will associate each of their data set name masks with the same action statement. It is 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 an exception. When a data set name mask with this exact value is processed by the Automation Server, a catalog search is not performed, but the associated action is triggered as if a new cataloged data set matching the mask has been located. Automation Server symbols for the data set name, and for the first qualifier of the data set name have values of the 8-byte string NULLFILE. Use the data set name mask of NULLFILE to trigger scheduled actions which do not depend on the creation of a particular data set.

Action statement syntax



action

FTP or JOB

template

Name of a member in the PARMLIB file.

TIME

This operand is optional, and the default is TIME(0000-2400), which specifies no time-of-day constraint.

hhmm-hhmm
Specifies a time-of-day range. Each *hhmm* value is four contiguous decimal digits that specify a time-of-day using the 24-hour clock. The minimum value is 0000 and the maximum value is 2400; the last two digits must not exceed 59. The two values are separated by a hyphen. Zero or more additional blanks are also permitted. The first *hhmm* specifies the time-of-day window start, while the second specifies the time-of-day window end. The window includes times which are after the window start and before the window end. However, if the second *hhmm* value is lower than the first, the window includes times which are after the window start **or** before the window end.

WEEK
This operand is optional. The default is WEEK(YYYYYYY), which specifies that the *action* can be run on every day of the week.

wkflags
Specifies a single contiguous 7-byte string, consisting of the uppercase characters Y or N. Each Y or N corresponds to a day of the week depending on its position in the string; the first corresponding to Sunday, the last to Saturday. If the character corresponding to a day of the week is N, the action is not processed on that day.

NOTB
This operand is optional. The default NOTB(1) specifies that the monthly window starts on the first possible day of the month. NOTB means "not before".

d1
Specifies a one or two digit decimal number in the 1-31 range. This number denotes the first possible day of the month on which the action is permitted.

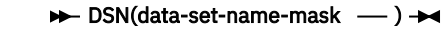
NOTA
This operand is optional. The default NOTA(31) specifies that the monthly window extends to the last day of the month. NOTA means "not after".

d2
Specifies a one or two digit decimal number in the 1-31 range. This number denotes the last possible day of the month on which the action is permitted.

MNTH
This operand is optional. The default value enables processing in every month of the year.

mthflags
Specifies a single contiguous 12-byte string, consisting of the uppercase characters Y or N. Each Y or N corresponds to a month of the year depending on its position in the string; the first corresponding to January, the last to December. If the character corresponding to a month of the year is N, the action is not processed in that month.

DSN statement syntax



DSN
Data set name.

data-set-name-mask
Specifies a data set name mask pattern which does not exceed 44 characters in length, and is used by 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. A double asterisk can be used to match a variable number of data set name qualifiers. The catalog search is restricted to entry type A non-VSAM data sets and entry type H generation data sets.

Time window considerations

You might normally specify the longest window duration that meets your scheduling requirements. Short window durations such as one or two minutes may not be reliable if the Automation Server has a large workload, or the system has resource shortages at that time. Long windows also increase the likelihood that retries can be scheduled on the same day. Alternatively, you might want to limit the time window duration to control the maximum number of automated retry attempts. Retry attempts occur at most once each 10-minute interval. For example, TIME(1420-1429) does not allow any time

to schedule a retry on the day of the original processing attempt, whereas TIME(1425-1435) allows 6 minutes for a retry on the same day if the original attempt fails in the first 5 minutes of the window. Assuming a failure occurs soon after the start of the time window, TIME(1400-1459) would allow up to 5 automated retry attempts

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 HSISJOB1, and are separately transferred to a z/OS system using the FTP commands in member HSISFTP1. All members are pointed to by the PARMLIB 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 an affinity to any system sharing the SPOOL. You do not need to run the job on the z/OS system where the Automation Server is running. The template name, HSISUIMP in this example, does not need to match the job name submitted by the Automation Server action.

Example 3:

A job stream stored in member WED2MNTH is to be submitted unconditionally on the second Wednesday of every month.

```
* RUN MONTHLY JOBSTREAM ON THE SECOND WEDNESDAY OF EVERY MONTH
 JOB ( WED2MNTH )   WEEK ( NNNYNNN )   NOTB ( 8 )   NOTA ( 14 )
 DSN ( NULLFILE )
```

Example 4:

A job stream stored in member NEWSHIFT is to run every day at 6:00 a.m, 2:00 p.m., and 10:00 p.m.

```
* RUN SHIFT TASK LIST REPORT AT THE START OF EACH SHIFT
 JOB(NEWSHIFT) TIME(0600-0630)
 DSN(NULLFILE)
 JOB(NEWSHIFT) TIME(1400-1430)
 DSN(NULLFILE)
 JOB(NEWSHIFT) TIME(2200-2230)
 DSN(NULLFILE)
```

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 called to process the record before it is written. You can use more than one symbol 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 symbols and symbols defined locally by the Automation Server. Most Automation Server local symbols are derived from the catalog entry data set name which, when discovered, triggers the instance of the action.

System symbols supplied by the operating system, as well as the &SMF and &SYSLPAR symbols supplied by the Automation Server, are available for use in the HSIAPARM member. The &SYSLPAR symbol will resolve to a null string if the system is running in a virtual machine.

Automation Server local symbols are provided in the following table:

Table 18. Automation Server local symbols

Symbol	Description
&SMF.	System SMF identifier.
&SYSLPAR.	System LPAR name.
&ASACTION.	Automation Server action name.
&ASKEY.	Automation Server control data set record key.
&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.

Example:

The data set triggering a JOB action is EXPUSER.IQ.ZIP. As a result, JCL DD statements referencing the data set in a template member can be represented as shown in this example:

```
//*-------------------------------------------------------------***
//* 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 JCL DD statements would be resolved by symbol substitution to:

```
DSN=EXPUSER.IQ.ZIP,DISP=SHR
```

This is the DSN= JCL statement output to the internal reader.

As symbol substitution is performed before the job is submitted, z/OS system symbols that cannot be used in batch job JCL, can be used in the Automation Server templates. The symbols are resolved using the system executing the Automation Server, which may not be the system where the submitted job executes.

Starting and stopping the Automation Server

When you install the Automation Server as a started task, you can run operator commands to start it and stop it. This task requires RACF security access.

Procedure

1. Assign RACF CONTROL access to the VSAM data set that is configured for the user ID that is assigned to the Automation Server.
2. Issue the system **START** command to start the Automation Server.
3. Issue the system **STOP** command to stop the Automation Server running as a started task or from running a batch job.

Excluding data sets

You can exclude data sets from Automation Server processing. To exclude a data set, it must have a record in the Automation Server control data set, with an indication in the record that the data set is already processed.

In the HSIAPARM member you define actions to be performed, and supply data set name masks specifying the data sets to be processed by the Automation Server. Data sets with these name patterns might already exist and have been processed before the Automation Server was implemented.

To exclude a data set, the name of the data must satisfy a selection mask pattern. To implement the exclusion, you can use the Automation Server data set name scouting program, HSIADSN. The HSISCUST post-installation job creates the HSIASSCT member in the JCLLIB data set. Run the HSIASSCT job to start the scouting program.

The program reads the HSIAPARM member, searches the catalog for every specified data set name mask, and writes a record for each data set that it discovers. The job then sorts the records into key order and copies them into the VSAM control data set. Every record loaded into the control data set in this way indicates a specific action with a status of complete.

If you want to continue processing some of the data sets, you can manually delete them from the sequential data set before the data is copied into the control data set.

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 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. This cleanup cycle consists of reading the ASCDS sequentially, and deleting records for data sets which have not been found by a recent catalog search. Records for data sets that have been found in the catalog in the current month or immediately prior calendar month are retained. The Automation Server issues a sysplex-wide exclusive enqueue in the HSIZNQAS queue to lock out the HSIAUTL program during the brief cleanup period.

As with most VSAM data sets with ongoing record insertion and deletion activity, it is advisable to periodically reorganize the ASCDS.

Automation Server Utility program

The Automation Server Utility program HSIAUTL can be used to extract and modify the contents of the VSAM ASCDS. It can be used as an ad hoc administrative tool or in production jobs to automate retry requirements.

When HSIAUTL is executed in jobs submitted by the Automation Server, the ASACTION and ASKEY symbols can be used to advantage. In particular, the ASKEY symbol can be used to refer to the job's own

ASCDS record, allowing it to be assigned the awaiting-retry status - usually in a step that is executed after it has been determined that the job has not been successful.

Here is an example of JCL used to execute the Automation Server Utility program:

```
//ASUTIL  EXEC PGM=HSIAUTL
//STEPLIB  DD  DISP=SHR,DSN=hlq.SHSIMOD1
//ASCDS    DD  DISP=SHR,DSN=opshlq.ASCDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
  MAKEPEND &ASKEY.
/*
```

Files used by the Automation Server Utility

Several files must be available for use by the Automation Server Utility.

STEPLIB
Load library containing the product software. Not required if IBM Z Software Asset Management is installed into the system linklist.

ASCDS
A VSAM KSDS control data set used by the Automation Server.

SYSPRINT
The output report file.

SYSOUT
Specifies the message report file for Language Environment.

SYSIN
The input file containing control statements.

Automation Server Utility control statements

Program HSIAUTL supports the following request types:

- **LISTALL** - list records without regard to the progress status.
- **LISTDONE** - list records pertaining to completed actions.
- **LISTPEND** - list records pertaining to incomplete or pending actions.
- **LISTRC** - list a specific record and issue a return code based on its status.
- **MAKEDONE** - convert a pending action record to a completed action record.
- **MAKEPEND** - convert a completed action record to a pending action record.

The following points apply when encoding requests in the SYSIN file:

- Blank records are processed as comments.
- The first non-blank data in a record must be either a request type from the above list or an asterisk.
- Records where the first non-blank is an asterisk are processed as comments.
- Requests must be completely encoded in the first 71 data bytes of the input record.
- Request continuations across multiple records are not supported.
- Request operands are deemed to start with the first non-blank character after the statement type.
- Request operands must be separated from the request type by one or more blanks.
- Request operands are deemed to end with the last non-blank character in the first 71 data bytes of the record.
- Request operands may contain embedded blanks.
- For all request types except LISTRC, the supplied operand will be treated as a generic key so that all records with keys that match up to the specified length can be processed by a single request.
- For LISTRC requests, the operand will be extended with blanks if necessary to form a full-length key.

- LISTRC requests require an operand.
- To prevent likely data loss, MAKEDONE and MAKEPEND requests require an operand.

The following points apply when executing Automation Server Utility requests:

- Valid generic requests do not set a non-zero completion code even if no matching records are found.
- LISTRC requests result in the following completion codes:

  0
  the record matching the specified key has a completed status.

  4
  the record matching the specified key has pending status.

  8
  there is no record which matches the specified key
- The HSIAUTL completion code is the highest result of all processed requests.
- When an invalid request is recognized, HSIAUTL will immediately terminate with a completion code of 12 without processing or reporting subsequent requests.

Each request is processed in isolation in the order supplied by the SYSIN file. Each LIST request will cause the ASCDS to be opened for input, and each MAKE request will cause the ASCDS to be opened for update. The user running the HSIAUTL program will need READ security access to the ASCDS to execute LIST requests and will need UPDATE security access to the ASCDS to execute MAKE requests. The HSIAUTL program does not delete records from or add records to the ASCDS, so CONTROL access is not required.

The ASCDS is serialized by sysplex-wide ENQs using the HSIZNQAS queue name. A shared ENQ is used for LIST requests and an exclusive ENQ is used for MAKE requests. This serialization scheme ensures that requests do not interfere with other execution instances of HSIAUTL or with the cleanup cycle of the Automation Server. The ASCDS shareoptions of (4,3) ensure that the Automation Server references refreshed data when using keyed direct access.

Automation Server Utility report details

Requests that process ASCDS records will report the contents of those records in SYSPRINT output. LIST requests will issue a report line for each matching record. MAKE requests will issue two report lines for each matching record with the first showing the record contents that were read, and the second showing the contents that were written back to the ASCDS. Altered records will be marked with an UPDATED indicator to the right of the record listing in the report.

Column headings used to describe record contents are as follows:

Column Heading	Description
ACTION	ACTION or template member name.
DATA SET NAME	Name of data set triggering this instance of the action.
ADD-DATE	The date that this record was added to the ASCDS.
ADD-TM	The time that this record was added to the ASCDS.
CAT-DATE	The date that the subject data set was last found to be cataloged. Until the subject data set is processed, this is the date of the most recent attempt to process the data set.
PUT-TM	The time of the most recent attempt to process the data set. This item is not updated after the subject data set has been processed.
MOV-DATE	The date that the subject data set was successfully processed. This item is 8 zeros for pending entries awaiting retry.
MOV-TM	The time that the subject data set was successfully processed. This item is initially blank. If this item is not blank when MOV-DATE is 8 zeros, then this record has been processed by a MAKEPEND request.

Column Heading	Description
TRYCT	The number of attempts to process the subject data set that have been made. If this item is zero for a record in completed status, the record was either marked complete by a MAKEDONE request or the record was constructed by the Scout Utility.

Local dates and times are used by the Automation Server when setting item values in ASCDS records. Dates are 8 numeric characters formed from a 4-digit year followed by a 2-digit month followed by a 2-digit day-of-month. Times are 6 numeric characters made from a 2-digit hour, a 2-digit minute-of-hour value and a 2-digit seconds-of-minute value, in that order. A blank is left between each display column in the report for readability, but in the ASCDS the data items abut. When forming a key to use as program input, make sure that any leading data set name part of the key starts immediately after the 8-byte action name. Embedded blanks are allowed in the key. For example, to list all records for data sets with names beginning with USER.DATA processed by the JOB1 and IQIMPORT actions, the following 2 input records could be used:

```
LISTALL JOB1    USER.DATA
LISTALL IQIMPORTUSER.DATA
```

Unconditionally scheduled actions

The Automation Server can initiate actions based only on a scheduled time window without first performing a catalog search to find particular data set catalog entries. This is achieved by specifying NULLFILE in the relevant DSN statement in the HSIAPARM member of the PARMLIB library. To track whether such an action has been initiated or not across potential Automation Server stoppages, a record for each such schedule is maintained in the ASCDS. To distinguish between different schedules of the same action the scheduling filters are also encoded within the DATA SET NAME item of the relevant records. For example, if an action was scheduled to occur once between 07:30 and 08:29 every day and once between 20:00 and 20:39 every day from Monday to Friday, the DATA SET NAME values of the two records would be:

```
NULLFILE 0730_0829 YYYYYYY 01YYYYYYYYYYYY31
NULLFILE 2000_2039 NYYYYYN 01YYYYYYYYYYYY31
```

You can use MAKEPEND to cause reruns of such actions to occur, but a rerun will only be triggered if the Automation Server processes the updated record before the time window governing that record closes for that day.

Automation Server Utility request type descriptions

LISTALL
Lists all records that match the specified key part. If no key part was specified, then all records in the ASCDS will be listed.

LISTDONE
Lists all records that match the supplied key part with a non-zero MOV-DATE value. If no key part was specified, then all records in the ASCDS with a non-zero MOV-DATE value will be listed.

LISTPEND
Lists all records that match the supplied key part with a zero MOV-DATE value. If no key part was specified, then all records in the ASCDS with a zero MOV-DATE value will be listed.

LISTRC
Lists the record that matches the supplied key and will set a completion code based on its status. A key must be specified for the request to be valid. If the record's MOV-DATE value is non-zero, a completion code of 0 will be set. If the record's MOV-DATE value is zero, a completion code of 4 will be set. If no record matching the specified key was found, a completion code of 8 will be set.

MAKEDONE
Sets the MOV-DATE and MOV-TIME items to the current date and time in all records that match the specified key part. The record contents from before and after the change will be listed, with the

after-change listing marked with an UPDATED indicator to the right of the record's data in the report. To reduce the likelihood of accidental data loss, a key part must be specified for the request to be valid.

MAKEPEND
Sets the MOV-DATE to zeros in all records that match the specified key part. MOV-TIME will not be updated. The record contents from before and after the change will be listed, with the after-change listing marked with an UPDATED indicator to the right of the record's data in the report. To reduce the likelihood of accidental data loss, a key part must be specified for the request to be valid.

HSIAUTL examples

In the following examples, SYS2.SHSIMOD1 is the name of the program library containing the HSIAUTL program, and PROD.ASCDS is the name of the Automation Server control data set.

Example 1

A job submitted on a regular basis by the Automation Server is to report all pending data sets that are also intended to be processed by the same JOB action.

The following job step is added to the end of the existing JCL template member in the PARMLIB library:

```
//BACKLOG EXEC PGM=HSIAUTL
//STEPLIB  DD  DISP=SHR,DSN=SYS2.SHSIMOD1
//ASCDS    DD  DISP=SHR,DSN=PROD.ASACDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
  * Report all unprocessed data sets for this data flow
  LISTPEND &ASACTION
/*
```

Example 2

All records currently present in the ASCDS are to be listed.

The following job step is run:

```
//DUMPALL EXEC PGM=HSIAUTL
//STEPLIB  DD  DISP=SHR,DSN=SYS2.SHSIMOD1
//ASCDS    DD  DISP=SHR,DSN=PROD.ASACDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
  * List all records
  LISTALL
/*
```

Example 3

All records pertaining to the UIMPORT action are to be listed

```
//UIMPLST EXEC PGM=HSIAUTL
//STEPLIB  DD  DISP=SHR,DSN=SYS2.SHSIMOD1
//ASCDS    DD  DISP=SHR,DSN=PROD.ASACDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
  * List all UIMPORT records
  LISTALL UIMPORT
/*
```

Example 4

The following data sets have all been processed by action DATAPROC successfully:

```
               HLQ.MYFILE
               HLQ.MYFILE.DATA1
               HLQ.MYFILE.DATA2
               HLQ.MYFILE.DATA3
               HLQ.MYFILE.DATA4
```

but HLQ.MYFILE needs to be reprocessed.

HLQ.MYFILE alone is set to awaiting-retry by the following job step:

```
//RETRY1  EXEC PGM=HSIAUTL
//STEPLIB  DD  DISP=SHR,DSN=SYS2.SHSIMOD1
//ASCDS    DD  DISP=SHR,DSN=PROD.ASACDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
 MAKEPEND DATAPROCHLQ.MYFILE
 MAKEDONE DATAPROCHLQ.MYFILE.
/*
```

Example 5

It is decided that a job submitted by the Automation Server should flag its own retry requirement if processing fails. Processing is deemed to have failed if any job step ends with a condition code greater than 4 or if any step abends.

Assuming local naming conventions prevent the existence of other data sets (processed by the Automation Server) with names that begin with the name of the data set being processed by the job, the following job step can be appended to the job's JCL template in the PARMLIB library:

```
//FAILURE EXEC PGM=HSIAUTL,COND=((4,GE),EVEN)
//STEPLIB  DD  DISP=SHR,DSN=SYS2.SHSIMOD1
//ASCDS    DD  DISP=SHR,DSN=PROD.ASACDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
  MAKEPEND &ASKEY
/*
```

Reporting with the Analyzer

The primary reporting facility in IBM Z Software Asset Management is the Analyzer.

The Analyzer runs as a started task or batch job on the same z/OS host as the Db2 Subsystem or SQLite database that contains the IBM Z Software Asset Management database(s).

The Analyzer has two modes:

Online mode
A PC Browser, for example Firefox, is used to communicate with the Analyzer for interactive queries.

Batch mode
This mode uses the Analyzer to generate the report to an output file. The Batch mode is useful when you want to automate reports or develop your own reports. Batch mode is also useful when you want to select multiple criteria, such as multiple libraries or multiple users which you cannot do online from some reports.

All Analyzer reports can be run in online and batch modes and can produce the following output formats:

- XML (xml)
- HTML (htm)

- Excel (Excel)
- Text line (txt)
- Comma Separated Value (csv)

Analyzer prerequisites

The Analyzer uses the Db2 Call Library Interface (ODBC/CLI), also used by the Inquisitor Import, Usage Import and other batch components, and the z/OS socket application programming interface. For the SQLite database, the Analyzer uses an internal ODBC interface.

There is no dependency on any other middleware components. For example, no dependency exists on the HTTP Server, WebSphere® Application Server, or Java™.

The Analyzer has been designed with minimal prerequisites. These are:

- V9.1 supports only V9.1. Repositories and depends on TPARAM parameter – e.g. BUILDLVL = 'V940D007'.
- The Analyzer must be run on the same z/OS host as the Db2 subsystem or SQLite database that contains the repositories.
- The user ID of the Analyzer address space must have previously been granted access to the databases. See the HSISGRNT job in the JCLLIB for sample JCL to grant access.
- When running the Analyzer in the online mode, you need access to a TCP/IP port. The default is port 9000.
- When running the Analyzer in online mode with SECURITY=SYSTEM, the Analyzer SHSIMOD1 load library must be defined to the z/OS Authorized Program Facility (APF). In addition, all data sets in the Analyzer STEPLIB, or JOBLIB DD concatenation, must be defined to APF.
- You can run the Analyzer in online mode while Inquisitor Import or Usage Import is also updating data into the repositories. However, the Analyzer reports may not display the correct information on the latest updates due to concurrency issues in Db2. To ensure that the latest correct information is displayed, do not run operational jobs that update data in the repositories while users are running reports with the Analyzer. This is not an issue for the SQLite database, as only single thread is allowed.

Analyzer JCLLIB and PARMLIB members

Several JCLLIB and PARMLIB members are used when you run the Analyzer to generate reports.

The members in the JCLLIB contain sample JCL to run the Analyzer.

Table 19. JCLLIB members for Analyzer

Member	Description
HSIJANLO	Analyzer PROC for online mode. Copy this PROC from the JCLLIB to a system PROCLIB data set to run the Analyzer as a started task
HSISANLB	Analyzer batch job for batch mode
HSISANLO	Analyzer batch job for online mode
HSISANS1	Define the Analyzer security profiles in RACF (only applicable for Analyzer SECURITY=SYSTEM setting)
HSISANS2	Generate the Analyzer SSL certificate in RACF (only applicable for Analyzer SECURITY=SYSTEM setting)
HSISANS3	Connect the Analyzer user ID to an existing SSL certificate in RACF (only applicable for Analyzer SECURITY=SYSTEM setting)

The following members in the PARMLIB contain sample configuration settings for the Analyzer in online mode. These members are referenced with the TPARAM setting in the HSIJANLO PROC.

Table 20. PARMLIB members for Analyzer	
Member	**Description**
HSISANP1	SECURITY=BASIC HTTP communications with basic security
HSISANP2	SECURITY=SYSTEM HTTPS (SSL encrypted) communications with z/OS system security (SAF/RACF) SECURITY=SYSTEM-TLS SSL encryption handled by AT-TLS configuration Refer to the HSISANS1, HSISANS2, and HSISANS3 members in JCLLIB for sample JCL to define RACF profiles/certificates.

Running the Analyzer in online mode

The primary reporting facility in IBM Z Software Asset Management is the Analyzer. You can use the Analyzer in online mode to view reports, run queries, and drill down to related reports.

About this task

Note: The V9.1 Analyzer will support only V9.1 repositories. Repositories from earlier releases are NOT supported and will not appear in the V9.1 Analyzer.

Note: Db2 only: It is recommended that not more than 50 repositories be defined in a Db2 subsystem that is referenced by each Analyzer. This is to prevent overload to the Analyzer and also for easier management of repositories.

Some Analyzer reports may require extensive searches and sorts. For performance reasons, increase the sizes of the tablespaces belonging to the work file database and, also consider using different definitions for the Bufferpool (instead of the default BP0 or BP32K).

Db2 only:

1. The Db2 subsystem must first be active before starting up the Analyzer. After the Analyzer starts, it needs to access the Db2 Catalogs to obtain the list of repositories. If the Db2 subsystem is not active, the error message below appears in the Analyzer job log.

```
13:15:09(0) COULD NOT QUERY SYSIBM.SYSTABLES FOR REPOSITORY -
SQLAllocHandle for connection handle failed - rc
-1
{DB2 FOR OS/390}{ODBC DRIVER}  SQLSTATE=58004  ERRLOC=2:56:9
  CAF "CONNECT" failed using DB2 system:DSN1
  RC=08 and REASON=00f30012
```

2. The Db2 subsystem can be stopped while the Analyzer is still active, but it will remain in a wait state. If a user logs in and tries to access a repository, the error message below will appear on the Analyzer report. When the Db2 subsystem is next started, users can login again and will automatically re-establish connection to Db2.

```
SQLAllocHandle for connection handle failed
 - rc -1
{DB2 FOR OS/390}{ODBC DRIVER}  SQLSTATE=58004  ERRLOC=2:56:9
  CAF "CONNECT" failed using DB2 system:DSN1
  RC=08 and REASON=00f30012
```

HSISANLO job in the JCLLIB is used to run Analyzer in online mode as a batch job.

```
//HSISANLO EXEC HSIJANLO,TPARAM=HSISANP1
```

To run the Analyzer in online mode as a Started Task, copy the HSIJANLO from the JCLLIB to a system PROCLIB data set. Here is a sample for running the started task in a Db2 subsystem.

```
//HSIJANLO PROC TPARAM='HSISANP1',       TPARAM input parms
//             CUST='HSISANCQ',
//             CLI='HSISCLI'
//*
//ANALYZER EXEC PGM=HSICANLZ,REGION=0M,TIME=NOLIMIT
//STEPLIB  DD DISP=SHR,DSN=&DB2EXIT
//         DD DISP=SHR,DSN=&DB2LOAD
//         DD DISP=SHR,DSN=&HSI..SHSIMOD1
//         DD DISP=SHR,DSN=&CEERUN
//         DD DISP=SHR,DSN=&CBCDLL
//SYSPRINT DD SYSOUT=*,LRECL=500
//HSIANL1  DD DISP=SHR,DSN=&HSI..SHSIANL1
//HSIANL2  DD DISP=SHR,DSN=&HSI..SHSIANL2
//HSICUST  DD DISP=SHR,
//       DSN=&HSIINST..&DB..PARMLIB(&&CUST)
//*HSINLS   DD DISP=SHR,DSN=&HSI..SHSIANL1(HSINLSJP)
//TPARAM   DD DISP=SHR,
//       DSN=&HSIINST..&DB..PARMLIB(&&TPARAM)
//DSNAOINI DD DISP=SHR,
//       DSN=&HSIINST..&DB..PARMLIB(&&CLI)
//WORK0    DD DISP=(NEW,DELETE),
//            UNIT=SYSALLDA,SPACE=(CYL,(100,50),RLSE)
//APPSTATS DD SYSOUT=*,LRECL=1000
//APPTRACE DD SYSOUT=*
```

When the Analyzer is run with online mode, configuration options must be defined in the TPARAM DD, including the communication port, security mode, and inactivity timeout.

Analyzer communication port

The Analyzer communication port is defined by using the HTTPPORT setting.

Both sample PARMLIB members HSISANP1 (BASIC security) and HSISANP2 (SYSTEM and SYSTEM-TLS security), have the following:

```
*********************************************************************
* HTTPPORT defines the TCP/IP port used for communications.          *
*                                                                   *
*   If HTTPPORT = 9000  is defined on a system with a TCP/IP host   *
*   called sys1.mycompany.com, to access the Analyzer the user      *
*   would specify the following URL in their PC Browser:            *
*        http://sys1.mycompany.com:9000  if SECURITY=BASIC          *
*    or https://sys1.mycompany.com:9000  if SECURITY=SYSTEM         *
*                                                                   *
*   The port specified must be available on your system.            *
*                                                                   *
*   TSO NETSTAT can be used to check if a port is available e.g.:   *
*   TSO NETSTAT (PORT 9000       --*  is port 9000 in use?         *
*   TSO NETSTAT PORTL(PORT 9000  --*  is port 9000 reserved?       *
*                                                                   *
*   If no entries are returned from these NETSTAT commands, the port  *
*   is most probably available.  At some sites, you may need your   *
*   Network Systems Programmer to reserve a port for FZVSAM Analyzer.  *
*                                                                   *
*********************************************************************
HTTPPORT = 9000
```

If HTTPPORT is not specified, or is set to 0, the Analyzer fails with message “COULD NOT OPEN DD:SYSIN”.

Analyzer inactivity timeout

The Analyzer communication inactivity timeout is defined by using the INACTIVITY_TIMEOUT setting.

Both sample PARMLIB members HSISANP1 (BASIC security) and HSISANP2 (SYSTEM and SYSTEM-TLS security), have the following:

```
*********************************************************************
* INACTIVITY_TIMEOUT defines the inactivity timeout in minutes.     *
```

```
*                                                                     *
*   The minimum value is 5 and the maximum is 9999.                   *
*                                                                     *
*   If INACTIVITY_TIMEOUT = 0 is defined, no timeout will occur.      *
***********************************************************************
```

If INACITIVITY_TIMEOUT is not specified, then it defaults to 30 minutes.

Analyzer security

You can view Analyzer reports in a web browser, such as Firefox, and you can communicate with the Analyzer utility to perform interactive queries.

Some of the Analyzer reports contain a large amount of information and it is recommended that you use a screen resolution of at least 1440 x 900 pixels to view them.

The following table describes the security modes that you can configure for accessing Analyzer online.

Table 21. Security modes for accessing Analyzer online

Security configuration	Communication mode	Access ID and password	Access permissions
SECURITY=BASIC	HTTP	Standard user ID and password. Default values are: • User: `fzvsamusr` and passwordFZVSAM • Admin: `fzvsamadm` and password FZVSAM	User ID `fzvsamusr` has limited access. User ID `fzvsamadm` has full access.
SECURITY=SYSTEM	HTTPS	z/OS system user ID and password Default: User TSO ID and password	Depends on the users' access to the HSICANLZ application and various TADzZ SW Asset Mgmt and Db2 Z SW Asset Mgmt.
SECURITY=SYSTEM-TLS	HTTPS	z/OS system user ID and password Default: User TSO ID and password	Depends on the users' access to the HSICANLZ application and various TADzZ SW Asset Mgmt and Db2 Z SW Asset Mgmt. An AT-TLS rule needs to be defined by a systems programmer.

Analyzer BASIC security

HSISANP1 in the PARMLIB defines basic user ID security settings for running the Analyzer.

User IDs FZVSAMADM and FZVSAMUSR can be used without any prior configuration. User ID AUID001 is a sample of how to restrict a user ID to certain databases.

```
***********************************************************************
* SECURITY=BASIC  - HTTP communications                               *
*                   with basic security defined in TPARAM DD below. *
*                                                                     *
***********************************************************************
SECURITY = BASIC

***********************************************************************
* The following settings are only applicable for                      *
* SECURITY=BASIC:                                                     *
*                                                                     *
*   AUTH_USER defines Userids and passwords for Analyzer logon        *
```

```
*   AUTH_DB   defines the repositories access                          *
*   AUTH_MENU defines the menus access                                 *
*                                                                      *
* The sample settings profile:                                        *
*  - FZVSAMADM userid:                                                  *
*       - Password FZVSAM                                              *
*       - Access to all repositories                                  *
*       - Access to all menu tabs                                     *
*  - FZVSAMUSR userid:                                                  *
*       - Password FZVSAM                                              *
*       - Access to all repositories                                  *
*       - Access to menu tabs ASSET, DISC + CUSTOM + ADMINR only      *
*  - AUID001 userid:                                                   *
*       - Password PW01                                               *
*       - Access to repositories AUDB01 + AUDB02 only                 *
*       - Access to menu tab ASSET only                               *
************************************************************************

*-----------------------------------------------------------------------
*AUTH_USER= USERID   , PASSWORD                                        *
*-----------------------------------------------------------------------
AUTH_USER = FZVSAMADM , FZVSAM
AUTH_USER = FZVSAMUSR , FZVSAM
AUTH_USER = AUID001  , PW01

*-----------------------------------------------------------------------*
*AUTH_DB  = REPOSITORY, LIST OF USERIDS AUTHORIZED TO SEE THE          *
*           REPOSITORIES                                               *
*-----------------------------------------------------------------------*
AUTH_DB   = *        , FZVSAMADM FZVSAMUSR
AUTH_DB   = AUDB01   , AUID001
AUTH_DB   = AUDB02   , AUID001

*-----------------------------------------------------------------------*
*AUTH_MENU= MENU_TAB , LIST OF USERIDS AUTHORIZED TO SEE THE MENU TAB *
*-----------------------------------------------------------------------*
AUTH_MENU = ASSET    , FZVSAMADM FZVSAMUSR AUID001
AUTH_MENU = DISC     , FZVSAMADM FZVSAMUSR
AUTH_MENU = ADMINR   , FZVSAMADM,FZVSAMUSR
AUTH_MENU = ADMIN    , FZVSAMADM
AUTH_MENU = CUSTOM   , FZVSAMADM FZVSAMUSR
* Activation/deactivation of the new Lib Classification items.
* ADMIN displays Admin menu with lib classification items deactivated.
*  (Must specify this to see the Admin menu at all)
* ADMIN(LIB_CLASSIFICATION) activates the lib classification items for the specified
users.
* AUTH_MENU = ADMIN(LIB_CLASSIFICATION) , FZVSAMADM
```

Analyzer SYSTEM and SYSTEM-TLS security

HSISANP2 in the PARMLIB defines the security settings for running the Analyzer.

The following SYSTEM and SYSTEM-TLS settings are defined:

```
************************************************************************
* FZVSAM Analyzer on-line mode settings for z/OS SYSTEM security and   *
* SYSTEM-TLS security                                                  *
************************************************************************
* SECURITY=SYSTEM - HTTPS (SSL encrypted) communications               *
*                     with z/OS system security (SAF/RACF).            *
*                     Refer to HSISANS1/2/3 in JCLLIB for sample JCL   *
*                     to define RACF profiles/certificates.            *
*                                                                      *
************************************************************************
* SECURITY=SYSTEM-TLS - SSL encryption handled by AT-TLS configuration*
*  NOTE: An AT-TLS rule needs to be defined by a system programmer.    *
* Following is an example of an AT-TLS rule:                           *
*  TTLSRule Secure_FZVSAM-Server                                         *
*  {                                                                   *
*    LocalPortRangeRef               port9133                        *
*    RemotePortRangeRef              portA11                         *
*    Direction                       Inbound                         *
*    TTLSGroupActionRef              gTLS                            *
*    TTLSEnvironmentActionRef        eFZVSAM                           *
*    Jobname                         SIMANL83                        *
*  }                                                                   *
************************************************************************
*                                                                      *
SECURITY = SYSTEM
```

```
*SECURITY = SYSTEM-TLS

*********************************************************************
* The following settings are applicable only for                    *
* SECURITY=SYSTEM:                                                  *
*                                                                   *
*   AUTH_HLQ defines SAF/RACF profile high level qualifier          *
*                                                                   *
*   AUTH_UPPERCASE=Y Analyzer will uppercase passwords when         *
*                    invoking SAF/RACF password authentication.     *
*                    When password phrase support has been          *
*                    enabled AUTH_UPPERCASE=Y has no effect, and    *
*                    mixed case is used.                            *
*   AUTH_UPPERCASE=N Analyzer will pass through mixed case passwords*
*                    when invoking SAF/RACF password authentication.*
*                                                                   *
*   GSK_KEYRING_FILE defines SAF/RACF Keyring name of SSL Certificate*
*   GSK_KEY_LABEL    defines SAF/RACF Label name of SSL Certificate *
*   GSK_....         defines optional z/OS SSL environment variables.*
*                    The z/OS Cryptographic Services Secure Sockets *
*                    Layer Programming manual explains the          *
*                    environment variables.                         *
*                    For example, define GSK_HW_CRYPTO = 32         *
*                    for SHA-256 digest generation.                 *
*                                                                   *
*********************************************************************
* The following settings are applicable only for                    *
* SECURITY = SYSTEM-TLS:                                            *
*   AUTH_HLQ         defines SAF/RACF profile high level qualifier  *
*                                                                   *
*   AUTH_UPPERCASE=Y Analyzer will uppercase passwords when         *
*                    invoking SAF/RACF password authentication.     *
*                    When password phrase support has been          *
*                    enabled AUTH_UPPERCASE=Y has no effect, and    *
*                    mixed case is used.                            *
*   AUTH_UPPERCASE=N Analyzer will pass through mixed case passwords*
*                    when invoking SAF/RACF password authentication *
*   Note: These 3 parameters are not required and must be commented  *
*         out when using SECURITY = SYSTEM-TLS.                     *
*     -GSK_KEYRING_FILE                                             *
*     -GSK_KEY_LABEL                                                *
*     -GSK_STATUS                                                   *
*                                                                   *
*********************************************************************
*                                                                   *
* JCLLIB(HSISANS1) contains sample JCL to define RACF profiles, using *
* a high level qualifier of 'FZVSAM'. If you have changed HSISANS1,   *
* you may also need to change the AUTH_HLQ TPARAM setting.          *
*                                                                   *
* JCLLIB(HSISANS2/3) contains sample JCL to define RACF SSL         *
* Certificates.  If you have changes HSISANS2/3, you may also need to *
* change the GSK_KEYRING_FILE and GSK_KEY_LABEL TPARAM settings.    *
*                                                                   *
*********************************************************************
AUTH_HLQ         = FZVSAM
AUTH_UPPERCASE   = Y
GSK_KEYRING_FILE = FZVSAM_KEYRING
GSK_KEY_LABEL    = FZVSAMCERT
GSK_STATUS       = OFF
```

HSISANS1 in the JCLLIB has sample JCL to define RACF security profiles.

Note: The RACF ID can be an existing RACF group (which user IDs have been connected to) and/or existing RACF user IDs.

If your z/OS system has been set up to use a third party alternative to RACF, you must define comparable settings in your third party security product.

```
/*-------------------------------------------------------------*/
/* FZVSAM ANALYZER DATABASE PROFILES                           */
/*-------------------------------------------------------------*/
 RDELETE FACILITY  FZVSAM.DB.AU*
 RDEFINE FACILITY  FZVSAM.DB.AU*          UACC(NONE)
 PERMIT            FZVSAM.DB.AU*          ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM,FZVSAMUSR,AUID001)

 RDELETE FACILITY  FZVSAM.DB.*
 RDEFINE FACILITY  FZVSAM.DB.*            UACC(NONE)
 PERMIT            FZVSAM.DB.*            ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM,FZVSAMUSR)
```

```
 PERMIT            FZVSAM.DB.*           ACCESS(NONE) -
   CLASS(FACILITY) ID(AUID001)
/*-------------------------------------------------------------*
/* FZVSAM ANALYZER MENU PROFILES                                *
/*-------------------------------------------------------------*
 RDELETE FACILITY  FZVSAM.MENU.ASSET
 RDEFINE FACILITY  FZVSAM.MENU.ASSET      UACC(NONE)
 PERMIT            FZVSAM.MENU.ASSET      ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM,FZVSAMUSR,AUID001)

 RDELETE FACILITY  FZVSAM.MENU.DISC
 RDEFINE FACILITY  FZVSAM.MENU.DISC       UACC(NONE)
 PERMIT            FZVSAM.MENU.DISC       ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM,FZVSAMUSR)

 RDELETE FACILITY  FZVSAM.MENU.ADMINR
 RDEFINE FACILITY  FZVSAM.MENU.ADMINR     UACC(NONE)
 PERMIT            FZVSAM.MENU.ADMINR     ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM,FZVSAMUSR)

 RDELETE FACILITY  FZVSAM.MENU.ADMIN
 RDEFINE FACILITY  FZVSAM.MENU.ADMIN      UACC(NONE)
 PERMIT            FZVSAM.MENU.ADMIN      ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM)

 RDELETE FACILITY  FZVSAM.MENU.ADMIN.LIB_CLASSIFICATION
 RDEFINE FACILITY  FZVSAM.MENU.ADMIN.LIB_CLASSIFICATION UACC(NONE)
 PERMIT            FZVSAM.MENU.ADMIN.LIB_CLASSIFICATION ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM)

 RDELETE FACILITY  FZVSAM.MENU.CUSTOM
 RDEFINE FACILITY  FZVSAM.MENU.CUSTOM     UACC(NONE)
 PERMIT            FZVSAM.MENU.CUSTOM     ACCESS(READ) -
   CLASS(FACILITY) ID(FZVSAMADM,FZVSAMUSR)

  SETROPTS RACLIST(FACILITY) REFRESH
```

SSL Certificates

When the Analyzer is running with SYSTEM=SECURITY, you must have an SSL Certificate defined in your SAF/RACF security system. You can either generate your own certificate, or connect to an existing certificate.

HSISANS2 in JCLLIB has sample JCL to generate SSL certificates in RACF.

```
//*********************************************************************
//*                                                                   *
//* To enable FZVSAM Analyzer to use HTTP secure (HTTPS) the following *
//* steps should be implemented by your site's RACF Administrator:    *
//* 1. Delete KEYRING(FZVSAM_KEYRING) and certificates with the        *
//*    labels FZVSAMCERT and LOCALCA.                                  *
//* 2. Activate RACF Classes required for digital certificates.       *
//* 3. Define Keyring FZVSAM_KEYRING.                                  *
//* 4. Generate certificate.                                          *
//* 5. Connect to Keyring.                                            *
//* 6. Refresh RACF Classes required for digital certificates.        *
//* 7. Permit access to the Facility Class profiles and refresh.      *
//*                                                                   *
//*                                                                   *
//* The following JCL demonstrates a sample implementation:           *
//* 1. Update all occurrences of "Userid-running-HSISANLO" to reflect *
//*    your FZVSAM HTTPS environment.                                  *
//*                                                                   *
//* Do not change the RACF keyring 'FZVSAM_KEYRING' or label          *
//* 'FZVSAMCERT' unless you update the corresponding values in Analyzer*
//* PARMLIB member HSISANP2 and restart the Analyzer STC/Job.         *
//*-------------------------------------------------------------------*
//RACFDEF    EXEC  PGM=IKJEFT01,DYNAMNBR=30
//SYSTSPRT   DD    SYSOUT=*
//SYSTSIN    DD    *
 PROF NOPREF

 RACDCERT DELETE(LABEL('LOCALCA')) CERTAUTH
 RACDCERT DELETE(LABEL('FZVSAMCERT')) ID(Userid-running-HSISANLO)
 RACDCERT ID(Userid-running-HSISANLO) DELRING(FZVSAM_KEYRING)

 SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)
```

```
 RACDCERT ID(Userid-running-HSISANLO) ADDRING(FZVSAM_KEYRING)

 RACDCERT ID(Userid-running-HSISANLO) CERTAUTH GENCERT -
 SUBJECTSDN( O('Your Organization')   -
 CN('Your Domain')                    -
 C('US')) TRUST                       -
 WITHLABEL('LOCALCA')                 -
 KEYUSAGE(CERTSIGN)

 RACDCERT ID(Userid-running-HSISANLO) GENCERT -
 SUBJECTSDN (CN('FZVSAMCERT')                  -
 OU('Your Dept.')                             -
 C('US'))                                     -
 WITHLABEL('FZVSAMCERT')                       -
 SIGNWITH(CERTAUTH                            -
 LABEL('LOCALCA'))

 RACDCERT ID(Userid-running-HSISANLO)         -
 CONNECT(ID(Userid-running-HSISANLO)          -
 LABEL('FZVSAMCERT')                           -
 RING(FZVSAM_KEYRING)                          -
 DEFAULT                                      -
 USAGE(PERSONAL))


 RACDCERT ID(Userid-running-HSISANLO)         -
 CONNECT(ID(Userid-running-HSISANLO) CERTAUTH -
 LABEL('LOCALCA')                             -
 RING(FZVSAM_KEYRING)                          -
 USAGE(CERTAUTH))

 SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
/*
//PERMIT     EXEC  PGM=IKJEFT01,DYNAMNBR=30
//SYSTSPRT   DD    SYSOUT=*
//SYSTSIN    DD    *
 PROF NOPREF

  RDEL FACILITY IRR.DIGTCERT.LIST
  RDEL FACILITY IRR.DIGTCERT.LISTRING

  RDEFINE FACILITY IRR.DIGTCERT.LIST  UACC(NONE)
  RDEFINE FACILITY IRR.DIGTCERT.LISTRING  UACC(NONE)

  PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY)     -
  ID(Userid-running-HSISANLO) AC(READ)

  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) -
  ID(Userid-running-HSISANLO) AC(READ)

  SETR RACLIST(FACILITY)
REFRESH


/*
```

HSISANS3 in JCLLIB has sample JCL to connect to existing SSL certificates in RACF.

```
//*********************************************************************
//*                                                                   *
//* To enable FZVSAM Analyzer to use HTTP secure (HTTPS) using an      *
//* existing CA certificate, 'Entrust Secure Server Root CA' in our   *
//* example, the following steps should be implemented by your site's *
//* RACF Administrator:                                               *
//*                                                                   *
//* 1. Delete KEYRING(FZVSAM_KEYRING) and certificate with the         *
//*    LABEL('FZVSAMCERT').                                            *
//* 2. Activate RACF Classes required for digital certificates.       *
//* 3. Define Keyring FZVSAM_KEYRING.                                  *
//* 4. Connect the existing CA certificate to the Keyring.            *
//* 5. Refresh RACF Classes required for digital certificates.        *
//* 6. Permit access to the Facility Class profiles.                  *
//*                                                                   *
//*                                                                   *
//* The following JCL demonstrates a sample implementation:           *
//* 1. Update all occurrences of "Userid-running-HSISANLO" to reflect *
//*    your FZVSAM HTTPS environment.                                  *
//*                                                                   *
//* Do not change the RACF keyring 'FZVSAM_KEYRING' or label 'FZVSAMCERT'
//* unless you update the corresponding values in Analyzer PARMLIB    *
```

```
//* member HSISANP2 and restart the Analyzer STC/Job.                *
//*-----------------------------------------------------------------*
//RACFDEF    EXEC  PGM=IKJEFT01,DYNAMNBR=30
//SYSTSPRT   DD    SYSOUT=*
//SYSTSIN    DD    *
 PROF NOPREF

 RACDCERT DELETE(LABEL('FZVSAMCERT')) ID(Userid-running-HSISANLO)
 RACDCERT ID(Userid-running-HSISANLO) DELRING(FZVSAM_KEYRING)

 SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)

 RACDCERT ID(Userid-running-HSISANLO) ADDRING(FZVSAM_KEYRING)

 RACDCERT ID(Userid-running-HSISANLO) GENCERT -
 SUBJECTSDN (CN('FZVSAMCERT')                  -
 OU('Your Dept.')                              -
 C('US'))                                      -
 WITHLABEL('FZVSAMCERT')

 RACDCERT ID(Userid-running-HSISANLO)         -
 CONNECT(ID(Userid-running-HSISANLO)          -
 LABEL('FZVSAMCERT')                           -
 RING(FZVSAM_KEYRING)                          -
 DEFAULT                                      -
 USAGE(PERSONAL))

 RACDCERT ID(Userid-running-HSISANLO)         -
 CONNECT(ID(Userid-running-HSISANLO) CERTAUTH -
 LABEL('Entrust Secure Server Root CA')       -
 RING(FZVSAM_KEYRING)                          -
 USAGE(CERTAUTH))

 SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
/*
/*
//PERMIT     EXEC  PGM=IKJEFT01,DYNAMNBR=30
//SYSTSPRT   DD    SYSOUT=*
//SYSTSIN    DD    *
 PROF NOPREF

  RDEL FACILITY IRR.DIGTCERT.LIST
  RDEL FACILITY IRR.DIGTCERT.LISTRING

  RDEFINE FACILITY IRR.DIGTCERT.LIST  UACC(NONE)
  RDEFINE FACILITY IRR.DIGTCERT.LISTRING  UACC(NONE)

  PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY)     -
  ID(Userid-running-HSISANLO) AC(READ)

  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) -
  ID(Userid-running-HSISANLO) AC(READ)

  SETR RACLIST(FACILITY) REFRESH
/*
```

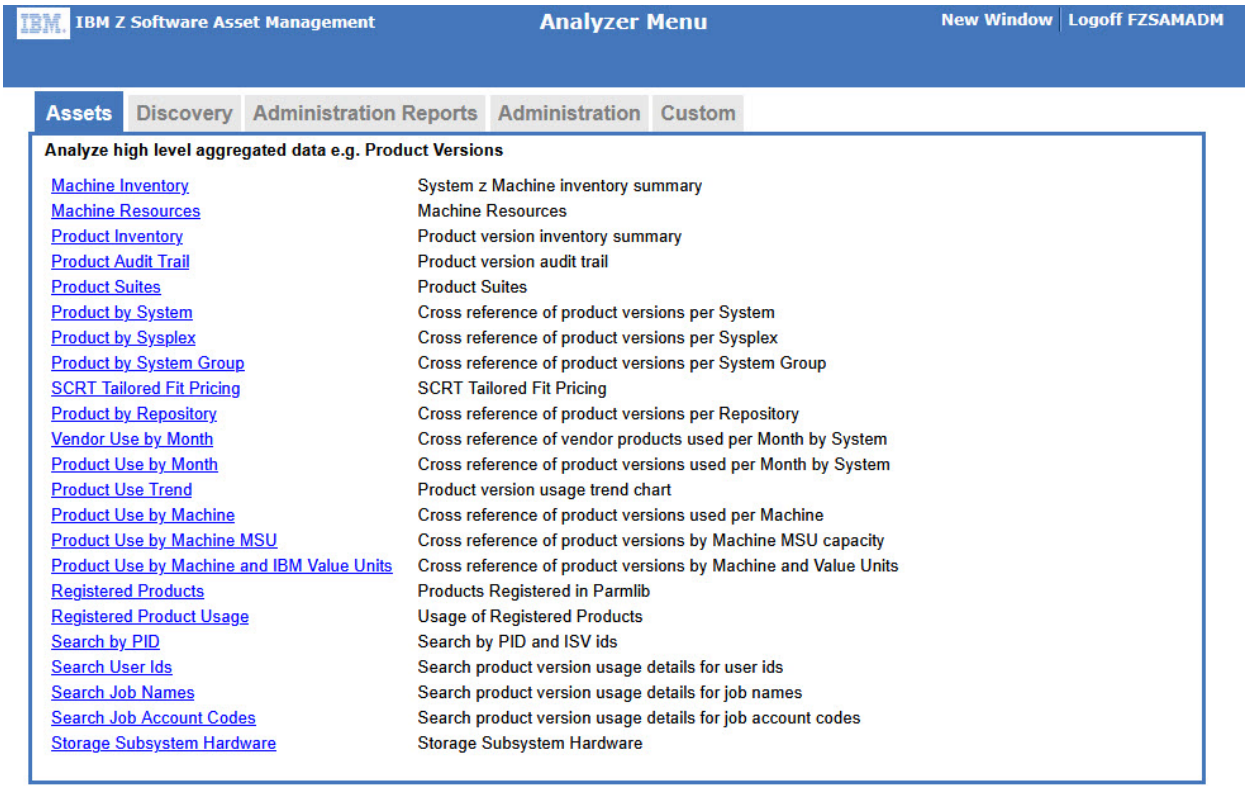
Online login to the Analyzer

With the Analyzer reporting utility, you can log in with a browser to gain access to the Analyzer Asset, Discovery, and Administration reports and to any Custom reports that you create.

To access the Analyzer online, enter the URL including the host name and port number, in the address bar of a browser, for example PTHOMU2.prod.hclpnp.com:9087PTHOMU3.prod.hclpnp.com:9000. In the Log In screen, provide the user ID and password associated with the BASIC security mode. See “Analyzer security” on page 131.

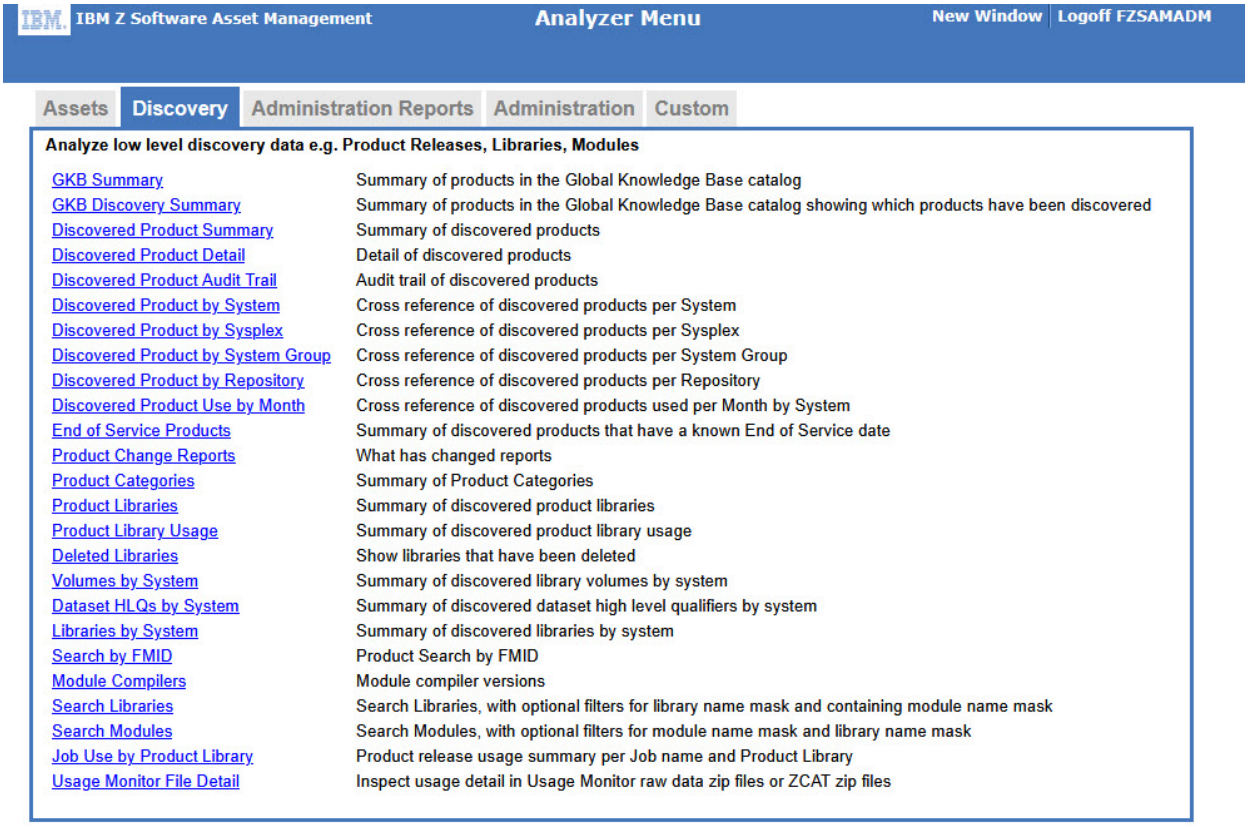
When you login to the Analyzer online, the **Analyzer Menu** window includes the following tabs:

- The **Assets** tab contains reports that query high level aggregated data, such as product versions. This level of data is useful if you are reconciling product licenses.

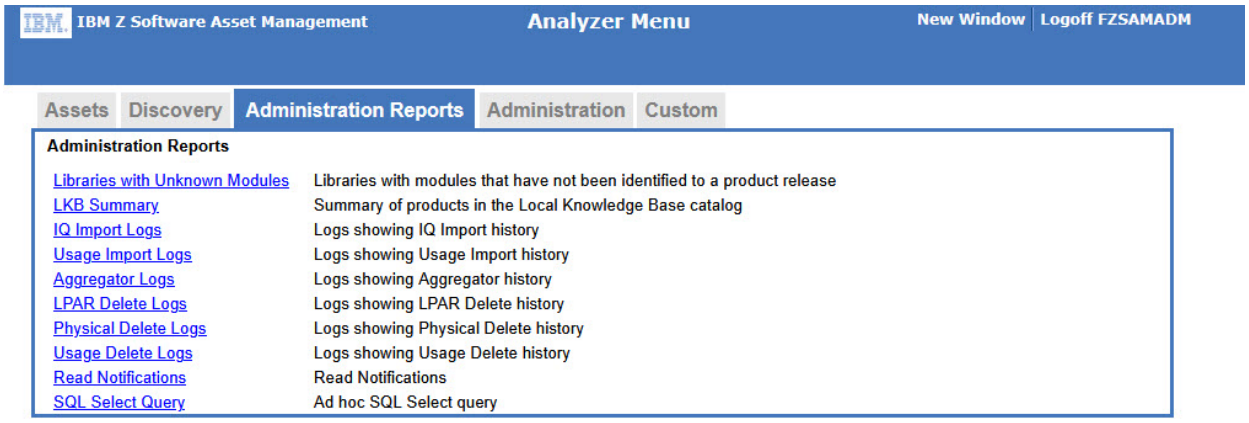


Screenshot of the **Assets** tab of the Analyzer online, including links to each Asset query.

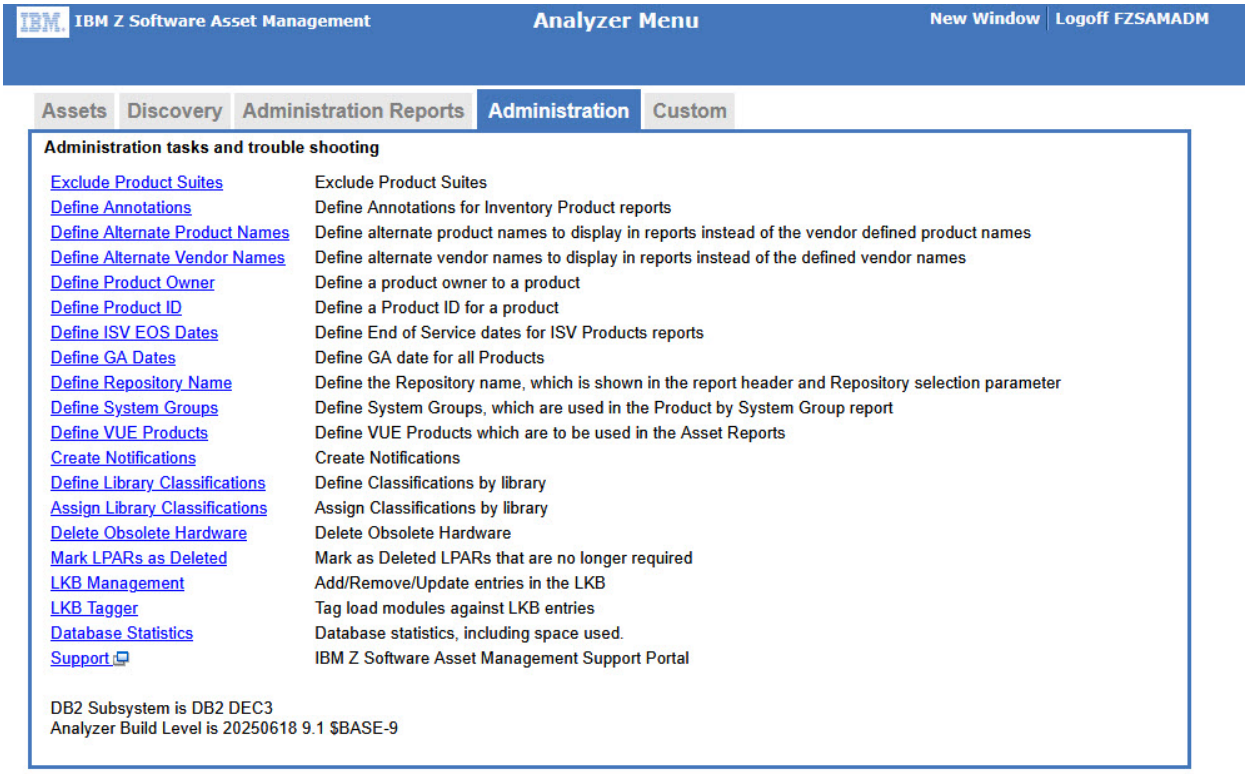
- The **Discovery** tab contains reports that query low-level discovery data, such as product releases, libraries, and modules. This level of data is useful if you support z/OS systems.Screenshot of the **Discovery** tab of the Analyzer online, including links to each Discovery query.



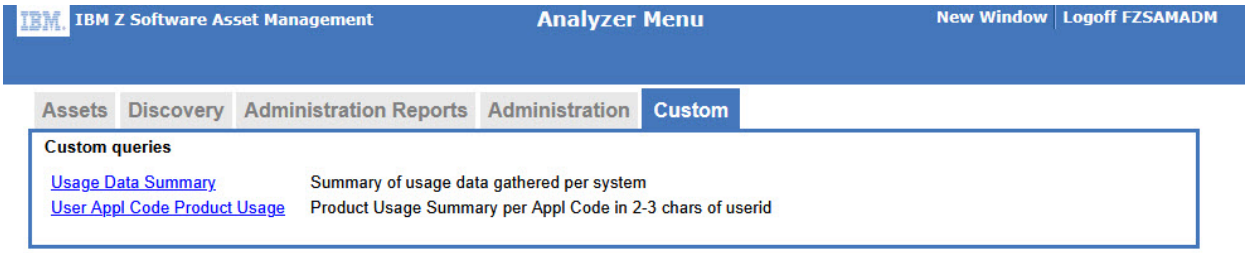
- The **Administration Reports** tab contains administration tasks where no updates are permitted.



- The **Administration** tab contains administration tasks and troubleshooting reports. These reports are designed for IBM Z Software Asset Management administrators and users only see this menu if they are granted specific access.



- The **Custom** tab contains your local custom reports. Two example custom reports are provided. Screenshot of the Custom tab of the Analyzer online, including links to the two custom queries that are provided with IBM Z Software Asset Management.



From any of the tabs, when you click the link to a report, the next window opens that contains parameter selection lists based on the data in your database. Select items in the parameter lists to construct a query. Hold down the Ctrl or Shift key to select multiple items from a list. When you have selected all required parameters, click **Submit** to run the query.

At the end of every report, the report name and parameters are shown in the same syntax that you can copy and paste into the HSISANLB batch job SYSIN DD deck to run the report in batch mode.

When you construct a query, if you choose the option `Output format and select Browser` as the output format, the report includes hyperlinks that you can use to drill down for more information.

You can download the content of a report, including the embedded content, in the following file formats:

- XML (xml)
- HTML (htm)
- Excel (Excel)
- Text (txt)
- Comma separated value (csv)

Controlling the Analyzer address space

The Analyzer supports the STOP, REFRESH, and TRACE z/OS modify commands.

Table 22. z/OS modify commands supported by the Analyzer

Command	Description

STOP	Stops the Analyzer address space. For example:
	`/F HSISANLO,STOP`
	You can also use the z/OS STOP command abbreviation:
	`/P HSISANLO`

REFRESH	Refreshes Analyzer report templates and NLS text. For example: `/F HSISANLO,REFRESH` This is typically used to load new Custom queries
TRACE	Toggles tracing on and off. For example: `/F HSISANLO,TRACE` This should only be used when requested by IBM Support.

Running the Analyzer in batch mode

If you want to automate report generation, you can run the Analyzer in batch mode.

Here is a sample for running the job in a Db2 subsystem.

HSISANLB in JCLLIB contains sample JCL.

```
//HSISANLB JOB ,'FZVSAM Analyz-Batch',REGION=0M,
//CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*JOBPARM SYSAFF=OMU2
//*MAIN   SYSTEM=(OMU2)
//*Running in Environment DBTYPE=DB2
//*
//*********************************************************************
//*  SET OUTFMT=XML     =  was previously "XLS"
**
//*  @headers           =  new optional parameter (see example below) **
//*  @headers = full    =  displays report title, timestamp, list     **
//*                           of parameters, and column headings      **
//*  @headers = title   =  displays report title, timestamp and       **
//*                         column headings
**
//*  @headers = none    =  displays column headings only
**
//*
**
//* FZVSAM Analyzer batch mode job:
**
//* a. Parameters are case sensitive
**
//* b. All parameters must start in column 1.
**
//* c. In OUTDSN, increase the size according to report output.       **
//* d. In WORK0, increase the size according to report output.        **
//*    WORK0 can be changed to a permanent dataset, if required.     **
//* e. In //TPARAM, setting TRACE = Y will force an Analyzer trace.   **
//*
**
//* Records are fetched and stored in the WORK0 file.                 **
//* Once the fetch is complete, records are then physically written   **
//* to the OUTDSN dataset, in 8K blocks.
**
```

```
//*                                                                    **
//*********************************************************************
//   JCLLIB  ORDER=(USERID.FZVSAM82.DC2.TF12V8C2.JCLLIB)
//*
//   SET OUTFMT=TXT
//*  SET OUTFMT=XLS
//*  SET OUTFMT=XML
//*  SET OUTFMT=CSV
//*  SET OUTFMT=HTM
//*
//   SET OUTDSN=&SYSUID..FZVSAMALZ.&OUTFMT  Output dsn
//*
//**DELOLD   EXEC PGM=IEFBR14
//**OUTDSN   DD DSN=&OUTDSN,
//**         DISP=(MOD,DELETE),UNIT=SYSALLDA,SPACE=(TRK,(0,0))
//*
//ALLOC    EXEC PGM=IEFBR14
//OUTDSN   DD DISP=(MOD,CATLG),DSN=&OUTDSN,
//            UNIT=SYSALLDA,SPACE=(CYL,(200,100),RLSE)
/*
//*********************************************************************
//ANALYZER EXEC PGM=HSICANLZ,REGION=0M,TIME=NOLIMIT
//STEPLIB  DD DISP=SHR,DSN=DB2VC10.DEC2.SDSNEXIT
//         DD DISP=SHR,DSN=DB2.VC10.SDSNLOAD
//         DD DISP=SHR,DSN=hlq.SHSIMOD1
//         DD DISP=SHR,DSN=CEE.SCEERUN
//         DD DISP=SHR,DSN=CBC.SCLBDLL
//SYSPRINT DD SYSOUT=*,LRECL=500
//HSIANL1  DD DISP=SHR,DSN=hlq.SHSIANL1
//HSIANL2  DD DISP=SHR,DSN=hlq.SHSIANL2
//HSICUST  DD DISP=SHR,
//       DSN= USERID.hlq.DC2.TF12V8C2.PARMLIB(HSISANCQ)
//*HSINLS  DD DISP=SHR,DSN=hlq.SHSIANL1(HSINLSJP)
//DSNAOINI DD DISP=SHR,
//       DSN=USERID.hlq.DC2.TF12V8C2.PARMLIB(HSISCLI)
//WORK0    DD DSN=&WORK0,DISP=(NEW,DELETE),
//            UNIT=SYSALLDA,SPACE=(CYL,(200,100),RLSE)
//TPARAM   DD DUMMY
//OUTPUT1  DD  DISP=OLD,DSN=&OUTDSN,LRECL=2000
//APPSTATS DD SYSOUT=*,LRECL=1000
//APPTRACE DD SYSOUT=*
//SYSIN    DD  *
/asset/audit_trail
vendor      = IBM
showfeature = on
multirep    = REPZ8
@headers    = full
/*
```

The report name and parameters are specified in the SYSIN DD and the output goes to the OUTPUT1 DD.

The simplest way to know what report name and parameters to specify is to run the report first using Analyzer in online mode. At the end of every report, the report name and parameters are listed in the syntax needed for batch mode. You can copy and paste this syntax into the batch SYSIN DD.

Alternatively, you can directly type in the parameters. Wildcard filters have been enabled to assist in this case.

REST API

REST API is an architectural style for building web applications that relies on HTTP methods. An application based on REST API communicates with the server by sending HTTP requests (such as GET, POST, PUT, DELETE) to specific URLs. The REST API provides access to Db2® repositories using common HTTP operations, enabling you to communicate with servers and exchange data.

The IBM Z Software Asset Management REST API is available only for Db2® repositories.

Prerequisites

- The installation process requires the following:

- Valid TSO userid. This userid must have Db2® authority to operate on repositories (from SYSOPR privilege, or higher).
- Have access to the UNIX System Services system
- Have permission to modify the APF authorisation bit
- Register a Rule with the Policy agent to make sure you have AT-TLS, as this is the only security schema supported. For more information, you can refer to the HTTPS section. Contact your system administrator before modifying any PAGENT rule as this is a OS related activity.
- HFS mounted with the SETUID options set to true.

Note: This REST API service supports IBM Z Software Asset Management 8.3 or higher.

Installing the REST API

This topic describes the steps to install REST API.

There is a small configurator that automatically generates both your `odbc.ini` and `api.config.toml`.It prompts for:

- The Db2 DSN
- The API listening port
- The desired log verbosity: suggested is info.
- Output destinations for each file: we suggest leaving it with the default values.
- Ibm_db_home: HLQ of the Db2 libraries.

Once you confirm, the tool writes a properly tagged ODBC/CLI file and a valid TOML configuration. This ensures your API server can be started immediately.

Name	Description	Default Value
Port	Port where the server will listen	8899
Database type	Type of database to use	Db2
DSN	DSN of the Db2 database	QXPOMU2DEC2
Log	Log level (DEBUG, INFO, WARN, ERROR, OFF)	INFO
ibm_db_home	HLQ of Db2 libraries.	Db2.VC10

The configurator will also generate the JCLs needed to run the API almost already configured.There is also inside the PAX a file called CONFIGURE.txt which describes the steps needed to run the configurator. You can find the same instructions in the following section.

Step by Step Configuration

1. Extract the pax file with `pax -rvf "//'<paxfile>'"` and position yourself into the created dist directory
2. Run **./bin/configurator** and follow the prompts. This will generate four files:
   - **odbc.ini**
   - **api.config.toml**
   - **hsisapie.jcl**
   - **hsisapip.jcl**
3. Take the generated jcls and copy them into a dataset of your choice. If you plan to copy the JCLs using option 3.17, you need to first convert them into the correct codepage for ISPF; This can be done by running the following commands while positioned inside the dist folder:

- iconv -f ISO8859-1 -t IBM-1047 hsisapip.jcl > hsisapip.converted.jcl
- iconv -f ISO8859-1 -t IBM-1047 hsisapie.jcl > hsisapie.converted.jcl

  Note: You can copy the converted JCLs via option 3.17.

4. Edit HSISAPIP such that the STDENV DD card points to the member where you copied HSISAPIE.
5. Submit HSISAPIP to start the API.

Expected Result

In the HSISAPIP generated JCL, you should obtain something similar to the following.

```
//HSISAPIP JOB, 'REST API job',REGION=0M,TIME=NOLIMIT,
// CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//* STEP1: Launch the REST API job
//STEPEXEC EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT
//STEPLIB DD DISP=SHR,DSN=db2.SDSNEXIT
//    DD DISP=SHR,DSN=db2.SDSNLOAD
//    DD DISP=SHR,DSN=db2.SDSNLOD2
//*
//STDIN  DD PATH='/dev/null',PATHOPTS=(ORDONLY)
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDENV DD DISP=SHR,DSN=&HSIINST.PARMLIB(HSISAPIE)
//STDPARM DD *
SH cd $UNX_API_PATH; ./bin/fzvsam-api/*
```

The HSISAPIE JCL should contain the following values:

PARAMETERS	DESCRIPTION
UNX_API_PATH	Path to the API installation folder (e.g. /api_install/dist).
IBM_DB_HOME	The hlq for Db2® libraries used for accessing SDSNC.H and SDSNMACS. If SDSNMACS uses a different prefix, then define Db2_MACS as an environment variable. If SDSNHC.H uses a different prefix, then define Db2_INC as an environment variable
DSNAOINI	The odbc `.ini` file contains ODBC/CLI parameters.
UNX_CONFIG_PATH	Path to the `toml` configuration file.
USE_PASSPHRASE	Whether to enable the use of case sensitive passphrases. See Authorization for more details.

In the logs you should see something similar to the following:

```
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] IBM Z Software
Asset Management 9.1 REST API - log file
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /end_of_service/count
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /end_of_service/report
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /product_inventory/count
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /product_inventory/report
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /product_use_by_machine/count
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /product_use_by_machine/report
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /product_use_by_month/count
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
route:  /product_use_by_month/report
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] GET Registered
```

```
route:  /repositories
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] POST
Registered route:  /any_query
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] Using DB2 with
local connection
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] DB2 DSN:
QXPOMU2DEC2
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] Hostname:
PTHOMU2
2025-05-16T05:17:00-04:00-[INFO]: analyzer[analyzer/internal/api.NewApiServices] API running on
port: 8111
```

Errors and Workarounds

The logs contain just an error of a failed connection to Db2: This is probably a result of a misconfigured odbc.ini file. This file is very sensitive to character tagging. Try modifying the character set with the **chtag** command. If you are still facing problems try creating a new odbc.ini file with the same content.

It is important to **not** put any space at the beginning of the JCL files.

The logs contain random gibberish: This is probably due to the fact that in your .profile you are running a bash shell that is performing character conversion. The api will save logs in ebcdic format. Remove bash from the .profile. Alternatively you can modify the STDPARM DD card of the HSISAPIP jcl by changing it from:

SH cd $UNX_API_PATH; ./bin/fzvsam-api -> PGM /full/path/to/bin/fzvsam-api

This will prevent the JOB from sourcing the .profile before execution.

Authorization

The API uses RACF as the authentication system. Credentials should be provided using the Basic scheme in each request. Basic Authentication (Basic Auth) is a simple authentication scheme built into the HTTP protocol. It works by sending the user credentials — a username and password, separated by ":" — encoded in Base64 within the Authorization header of each HTTP request.

The configurator will prompt whether to have the RACF service case sensitive or not. If is set as case sensitive then credentials are passed to racf as is without modifying, allowing for use of passphrases up to 256 characters. If instead is set a non case sensitive, then both username and password are turned into uppercase characters, allowing for a password of at most 8 characters.

Errors and Workarounds

In the logs each call to RACF returns a 20 error code: This means that the service executable used to query RACF is not APF Authorized. In order to fix this problem you can run extattr +a bin/racfexec from your installation folder. To verify that the APF attribute is set, run ls -E bin/from the installation directory, and check that the a attribute is set.

If the attributes is set but still see missing APF authorization verify that the HFS where the API reside is mounted with the SETUID options set to true.

HTTPS

The API is deployed using HTTP, and is recommended to use AT-TLS to secure connections. AT-TLS is a feature on z/OS systems that delegate to the operating system the task of securing TCP connections. In order to set up AT-TLS contact a system administrator or programmer. This usually involves defining policy rules in the Policy Agent (PAGENT). In this way the API can be started without directly providing key and certificates but it will ask the Operating system instead to use the key, certificate pair as defined in the PAGENT.

Graceful Server Shutdown Via SDSF

The REST server can be gracefully terminated from an SDSF session using standard operator commands. This ensures a clean shutdown, releasing all resources and stopping any ongoing activity without abrupt termination.

The following commands are supported:

- `/P <job_name>` - Issues a STOP command to terminate the job.
- `/F <job_name>,S` - Sends a command to the job requesting a graceful shutdown.
- `/F <job_name>,s` - Alternative syntax (equivalent to the above on z/O).

End Points

In this section, let us discuss the end points in detail.

End of Service

Name	Type	Description	Default Value	Required
repository	string	Repository name	“”	Yes
vendor	String	Name of vendor	“”	No
showsysname	boolean	Show system names	“”	No
showplxname	boolean	Show sysplex names	“”	No
shownoneos	boolean	Include non-EOS products	“”	No
start_date	*DateTime	Start date - with format: YYYY-MM-DD, YYYY- MM-DD HH:MM:SS	“”	No
end_date	*DateTime	End date - with format: YYYY-MM-DD, YYYY- MM-DD HH:MM:SS	“”	No
opsys	string	Whether to use the GKB or GKU. Empty for None	“”	Yes
showsysname	bool	Show system names	false	No
shownoneos	bool	Include non-EOS products	false	No
showplxname	bool	Show sysplex names	false	No

```
curl pthomu2.prod.hclpnp.com:8889/end_of_service/count?repository=<REPO>&start_date=2024-04-
05&end_date=2024-04-12&opsys=GKB
-X GET -u username:password
```

```
curl pthomu2.prod.hclpnp.com:8889/product_inventory/report?repository=<REPO>-X
GET -u username:password
```

Products Inventory

Table 23. Parameters				
Name	**Type**	**Description**	**Default Value**	**Required**
repository	string	Repository name	“”	Yes
vendor	string	Product vendor	“”	No
product	string	Product name	“”	No
system	string	System name	“”	No
product	string	Product name	“”	No
show_product_discovery_name	boolean	Show product discovery name	false	No
show_subcap_values	boolean	Show sub cap values	false	No
show_sysplex_names	boolean	Show sysplex names	false	No
show_system_names	boolean	Show system names	false	No
show_product_suites_only	boolean	Show product suites only	false	No

For example:

```
curl https://pthomu1.prod.com:8000/api/products/inventory/count?repository=<REPO>
-X GET-u username:password
```

```
curl pthomu2.prod.hclpnp.com:8889/product_inventory/report?repository=<REPO>-X
GET -u username:password
```

Product Use by Machine

Name	**Type**	**Description**	**Default Value**	**Required**
vendor	string	Product vendor	“”	No
product	string	Product name	“”	No
machine	string	Machine name	“”	
repository	string	Repository name	“”	Yes
start_date	*DateTime	Start date - with format: YYYY-MM-DD, YYYY-MM-DD HH:MM:SS	“”	No

Name	Type	Description	Default Value	Required
end_date	*DateTime	End date - with format: YYYY-MM-DD, YYYY-MM-DD HH:MM:SS	“”	No
show_product_discovery_name	Boolean	Show the product discovery names	false	No
show_s_s_pid	Boolean	Show SSPID values	false	No
show_hardware_partition	Boolean	Show the hardware partition	false	No
metrics: • LAST_USE_DATE • FIRST_USE_DATE • MODULE_EVENTS • PRODUCT_USE_BY_MONTH • USER_ID_COUNT • JOB_NAME_COUNT • SCRT_MSU	ProductUseBy MachineMetrics	Metrics	“”	No

For example:

```
curl pthomu2.prod.hclpnp.com:8889/product_use_by_machine/count?repository=<REPO>
-X GET -u username:password
```

```
curl pthomu2.prod.hclpnp.com:8889/product_use_by_machine/report?repository=<REPO>
-X GET -u username:password
```

Product Use by Month

Name	Type	Description	Default Value	Required
vendor	String	Product vendor	“”	No
product	String	Product name	“”	No
system	String	System name	“”	No
start_date	*DateTime	Start date - with format: YYYY-MM-DD, YYYY-MM-DD HH:MM:SS	“”	No

Name	Type	Description	Default Value	Required
end_date	*DateTime	End date - with format: YYYY-MM-DD, YYYY-MM-DD HH:MM:SS	“”	No
show_product_discovery_name	Boolean	Show product discovery name	false	No
show_s_s_pid	Boolean	Show SSPID values	false	No
metrics: • MODULE_EVENTS • PRODUCT_USE_BY_MONTH • USER_ID_COUNT • JOB_NAME_COUNT • JOB_ACCOUNT_COUNT • SCRT_MSU	Product Use By Machine Metrics	Metrics	“”	No
repository	String	Repository name	“”	Yes

For example:

```
curl pthomu2.prod.hclpnp.com:8889/product_use_by_month/count?repository=<REPO>
-X GET -u username:password
```

```
curl pthomu2.prod.hclpnp.com:8889/product_use_by_month/report?repository=<REPO>
-X GET -u username:password
```

Discovered Product Detail

Name	Type	Description	Default Value	Required
repository	string	Repository name	“”	Yes
start_date	string	Start date - with format: YYYY-MM	“”	No
end_date	string	Start date - with format: YYYY-MM	“”	No
system	string	System name	“”	No
vendor	string	Vendor name	“”	No
product_release	string	Product release name	“”	No
show_only_zos	bool	Only show z/OS Products	false	No
show_only_unx	bool	Only show UNIX Products	false	No
show_only_unknown	bool	Only show UNKNOWN Release	false	No

Name	Type	Description	Default Value	Required
exclude_unknown	bool	Exclude Unknown Releases	false	No

For example:

```
curl pthomu2.prod.hclpnp.com:8889/discovered_product_detail/count?repository=<REPO>
-X GET -u username:password
```

```
curl pthomu2.prod.hclpnp.com:8889/discovered_product_detail/report?repository=<REPO>
-X GET -u username:password
```

Running the utilities provided with IBM Z Software Asset Management

IBM Z Software Asset Management provides utilities that you run to perform routine functional tasks to maintain the product lifecycle.

Condensing usage data with the ZCAT utility

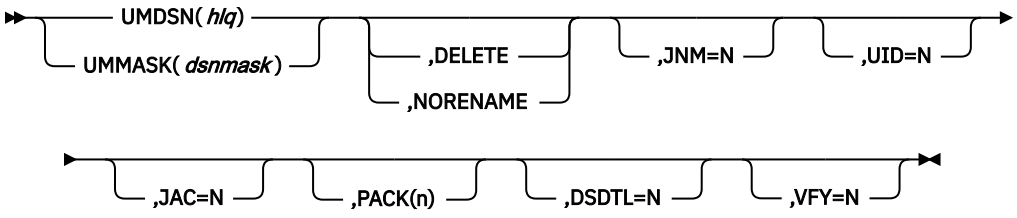
The ZCAT utility concatenates and condenses Usage Monitor data sets and generates a file that is then processed by the Usage Import program. When you condense the data produced by the Usage Monitor program, you can save storage space and improve the performance of the Usage Import program.

The Usage Monitor started task produces at least one usage data set per day. You can design a work flow that runs the ZCAT utility on the data sets on a weekly, fortnightly, or monthly basis before the Usage Import program processes them. Running the ZCAT utility on a weekly basis is useful, but depends on the amount of data that is produced and processed at your site. The Usage Monitor program collects detail about which job, account ID, and user ID are using each module of a particular library on a specified date. This information is output into multiple files that are produced on a daily basis. The ZCAT utility condenses the files in the following manner:

- Usage data across multiple files is condensed to a monthly granularity, as are the records stored in the Repository database.
- Redundant records in files and records that are not stored in the database, are omitted.
- Optionally, condensation can apply to user IDs, job names, or account ID details.
- The ZCAT output file is compressed and ready to be transmitted for Usage Import processing.

The following diagram shows the syntax of program parameters to run the ZCAT utility.

ZCAT utility syntax



Catalog search parameters

UMDSN and UMMASK are mutually exclusive. One must be specified if the ZCAT0001 DD is not allocated.

UMDSN(*hlq*)
hlq is the Usage Monitor data set high-level qualifier. When the **UMDSN** parameter is specified, ZCAT concatenates all data sets having names of hlq.Dyyyyddd.Thhmmsst where yyyyddd and hhmmsst are the timestamp patterns of data sets produced by the Usage Monitor. The *hlq* can contain wildcard characters of percent or asterisk. The percent character denotes a single character mask, and the asterisk character denotes all characters. For example UMDSN(hlq.**)would search for all data set names of hlq.**.D%%%%%%%.T%%%%%%%.

UMMASK(*dsnmask*)
dsnmask is the full dsn mask search criteria. It can be used to search for a pattern of files that differ from the files produced by the Usage Monitor. This parameter is useful if the files produced by the Usage Monitor have been renamed, but still need processing. Specifying UMMASK(hlq.D%%%%%%%.T%%%%%%%) is equivalent to specifying UMDSN(hlq)

Note: An easy way to remember the difference between UMDSN and UMMASK is to remember that UMDSN can accept the data set name prefix value specified in the Usage Monitor DSN setting, whereas UMMASK requires a mask which will match the entire data set name.

Input data sets found by searching the catalog may be zipped or unzipped. If zipped, then records before the first Usage Monitor header record will be discarded. If unzipped, the data set will not be processed unless the first record is a Usage Monitor header record.

Data set disposition parameters

One or more optional parameters can follow the mandatory parameters.

DELETE
Delete the input data sets after the output data set is successfully generated. **NODELETE** is the default.

NORENAME
Do not rename input data sets from hlq.D*.T* to hlq.D*.S* after the output data set is successfully generated. The default is to rename these input data sets to stop them being reprocessed by the ZCAT utility. Use this option only to rename the data sets before further ZCAT processing. This option stops double counting of usage data. This parameter is automatically set when UMMASK is used.

RENAME must not be explicitly specified with **DELETE.**

Data sets allocated to the ZCAT0001 DD are not included in **RENAME** and **DELETE** processing.

Optional condensation parameters

Improvements in performance and data storage space are gained by using the ZCAT utility options to carry out further condensation of data, ignoring data differences that are not important at your site, and do not appear in your regular reporting. You can still point the Usage Monitor File Detail Report to the saved archive of the Consolidated detail file (ZCATDETL), or to the Usage Monitor output files. ZCATDETL is produced by the ZCAT utility.

JNM

JNM is used to condense data based on job names.

JNM=N - Condense different job names to generic names of -STC-, -JOB-, -TSO- or -SYS-

JNM=Y - Preserve collected job name.

The shipped version of the HSISZCAT sample job specifies Y.

UID

UID is used to condense data based on user IDs.

UID=N - Replace collected user identifiers with blanks.

UID=Y - Preserve collected user identifiers.

The shipped version of the HSISZCAT sample job specifies Y.

JAC

JAC is used to condense data based on job account codes.

JAC=N - Replace collected job account codes with blanks.

JAC=Y - Preserve collected job account codes.

The shipped version of the HSISZCAT sample job specifies Y.

Note: The ZCATDETL file can be used to collect all valid importable input records into a single data set for archiving purposes, with the exception that duplicate user records are suppressed, and all user records are discarded if **UID=N** is specified.

Optional control parameters

DSDTL

DSDTL is used to control data set statistics reporting.

DSDTL=Y - Report data set condensation statistics to SYSPRINT.

DSDTL=N - Suppress the reporting of data set condensation statistics.

VFY

VFY is used to control whether the ZCATOUT file is to be verified after creation.

VFY=Y - After the ZCATOUT file is complete, it will be unzipped and read to verify that its contents are readable and that the expected number of records are present. This is the default.

VFY=N - Bypass ZCATOUT verification processing.

PACK

PACK is used to specify the zip compaction level used when writing zipped data.

PACK=n - where *n* is a decimal digit in the 0 to 9 range.

PACK=0 - Specifies that the ***shrink*** zip algorithm is used while higher values specify the compaction level of the ***deflate*** zip algorithm to be used. Higher compaction levels will achieve greater data compression, but will also consume disproportionately more CPU time.

PACK=1 - Is the default setting which requests the fastest level of the ***deflate*** method.

DD statements

SYSPRINT
Specifies the report file required by ZCAT which is usually allocated to SYSOUT. By default, RECFM=VBA and LRECL=137 will be used, though these can be overridden within some limits.

ZCATOUT
Specifies the name of the ZCAT output data set. This data set can then be used as the input to the Usage Import program, where usage details are imported into the database. If the ZCATOUT DD card is omitted, ZCAT by default writes to a data set having the name hlq.Dyyyyddd.Uhhmmsst (U instead of T implied by the high level qualifier (hlq) option for input data sets), where yyyyddd and hhmmsst refer to the date and time timestamp of the first processed input data set. If dynamically allocated, SPACE=(TRK,(768,255),RLSE) is used.

ZCATDETL
If the ZCATDETL DD is allocated, the uncondensed data is written to this data set. This allows detailed job name, user ID and job account information to be retained for subsequent analysis and/or reference. Any diagnostic records and records that fail validity testing are not written. Duplicate user records are suppressed. If UID=N is specified then all user records are discarded.

The ZCATDETL and ZCATOUT data sets are written as compressed (zipped) data by the ZCAT utility. If either of these is allocated to a data set with SMS compression a S213-C8 OPEN abend will occur. This abend will be trapped by ZCAT causing it to write uncompressed (unzipped) records which will

then be compressed by SMS. In this case, if VFY=Y is specified or defaulted to, the following message will be issued:

```
Output not zipped so skipping ZCATOUT verification
```

ZCAT0001

If the ZCAT0001 DD is allocated, it specifies one or more usage data zip archives to be processed by ZCAT. ZCAT0001 is processed after any data sets located by searching the catalog, and allows administrators to manually process specific data sets which may have fallen outside the usual processing regime, or may not fit any convenient data set name mask. Like dynamically allocated input data sets, data set(s) allocated to ZCAT0001 may be zip archives or may contain unzipped data. However, ZCAT0001 is treated by ZCAT as a single file, and a concatenation containing both zipped and unzipped data is not allowed.

```
//ZCAT EXEC PGM=HSIZCAT,PARM='UMDSN(hlq.**)'
//STEPLIB DD DISP=SHR,DSN=hsi.SHSIMOD1
//ZCATOUT DD DSN=&SYSUID..hlq.ZCATOUT,
// DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(50,50),RLSE),
// DCB=(DSORG=PS,RECFM=VB,LRECL=27994,BLKSIZE=27998)
//ZCATDETL DD DSN=&SYSUID..hlq.ZCATDETL,
// DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(CYL,(50,50),RLSE),
// DCB=(DSORG=PS,RECFM=VB,LRECL=27994,BLKSIZE=27998)
//SYSPRINT DD SYSOUT=*,HOLD=YES
```

In this example, all data sets having names of hlq.*.*.D%%%%%%%.T%%%%%%% are processed due to the **UMDSN** parameter. The condensed output is written to &SYSUID..hlq.ZCATOUT, where the SYSUID system symbol is the user ID of the person submitting the job. This file is then transmitted for Usage Import processing. All valid records are written to the ZCATDETL DD card, &SYSUID..hlq.ZCATDETL, which is then archived for reference purposes.

Deleting usage data with the Usage Deletion utility

Use the Usage Deletion utility to delete detailed, summarized, and aggregated usage data for a specified period for all systems in the repository. Each time you run the utility, usage data is aggregated to update the asset tables.

To minimize space utilization and improve SQL query performance, keep no more than three months of detailed module usage data and 13 months of aggregated product usage data.

If you do not run the Usage Deletion utility for some time, select a period of a few months, in order to keep the run times down to a reasonable time.

In the previous release, audit records from the TLOGUI table are also deleted by this utility. In the current release, these audit records are no longer deleted by this utility.

An audit table captures information of this utility when it is run, and these audit records can be browsed using an Analyzer report. As part of housekeeping, obsolete audit records from this utility are deleted by the MDEL Deletion utility, together with audit records from other utilities.

Running the Usage Deletion utility

To run the Usage Deletion, use the job HSISUDEL, in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

TPARAM parameters

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

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

KEEPDETAIL=
Default is 2. Number of months prior to the current month for which detailed and summarized module usage data are kept. KEEPDETAIL=0 means all detailed and summarized module usage data excluding those from the current month are deleted.

KEEPAGGR=
Default is 12. Number of months prior to the current month for which aggregated product usage data are kept. KEEPAGGR=0 means all aggregated product usage data, excluding those from the current month are deleted.

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.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

SID=
System Identifier of system for which usage should be deleted. Specify SID=ALLSIDS to delete usage data for all SIDs.

Note: 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.

Deleting a specific system with the System Deletion utility

Use the System Deletion utility to delete discovery, usage, and hardware data for a specified system.

For example, you can also use the System Deletion utility to delete a system that was accidentally imported into the repository, or to delete a system that is decommissioned.

An audit table captures information of this utility when it is run, and these audit records can be browsed using an Analyzer report. As part of housekeeping, obsolete audit records from this utility are deleted by the MDEL Deletion utility, together with audit records from other utilities.

Running the System Deletion utility

To run the System Deletion, use the job HSISLDEL, in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

TPARAM parameters

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository qualifier. Name of qualifier is &*REPZSCHM*.

SID=
System Identifier of system to be deleted.

Deleting obsolete data with the Physical Deletion utility

Use the Physical Deletion utility to delete obsolete data that are no longer required.

Over time, data that have been collected are either superseded or no longer valid, thus taking up space in the repository. Each time, you run this utility, obsolete data for the specified period are physically deleted for all systems in the Repository.

To keep the run time down to a reasonable time, set the date range for a smaller period.

An audit table captures information of this utility when it is run, and these audit records can be browsed using an Analyzer report. As part of housekeeping, obsolete audit records from this utility are deleted by the MDEL Deletion utility, together with audit records from other utilities.

This utility deletes obsolete records from the following tables:

- TLIBRARY
- TLIBSYS
- TMODULE
- TCSECT
- TPOVLIB
- TPRODUCT
- TVENDOR
- TVERSION

Running the Physical Deletion utility

To run the Physical Deletion, use the job HSISPDEL, in the JCLLIB. This job is generated from the HSISCUST post-installation customization job. Refer to comments in the job on how to run an SQL statement to list date ranges for deletion.

TPARAM parameters

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

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

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.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

RETENTION=

Number of months prior to the current month for which libraries marked as deleted are still retained. For example, if RETENTION=12, then only libraries marked as deleted and older than 13 months are physically deleted.

For a setting of RETENTION=0, only libraries marked as deleted for the current month are retained.

Deleting obsolete audit records with the MDEL Deletion utility

Use the MDEL Deletion utility to delete obsolete audit records that are no longer required.

During IQ Import, Usage Import and housekeeping utilities, audit records are collected. Over time, these obsolete records need to be deleted if they are no longer referenced. This utility deletes obsolete records from the following audit tables:

- TLOGIQ - audit records from IQ Import.
- TLOGUI - audit records from Usage Import.
- TLOGAGGR - audit records from Aggregator jobstep.

- TLOGLDEL - audit records from LDEL (System) Deletion.
- TLOGPDEL - audit records from Physical Deletion.
- TLOGUDEL - audit records from Usage Deletion.

In addition, this utility also deletes obsolete records from these two tables:

- TCHANNEL_PATH - obsolete records from IQ Import.
- TCONTROL_UNIT - obsolete records from IQ Import.

Running the MDEL Deletion Utility

To run the MDEL Deletion, use the job HSISMDEL, in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

TPARAM parameters

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

RETENTION=
Default is 24. Number of months prior to the current month for which audit records are kept. For a setting of RETENTION=0, audit records only for the current month are retained.

SID=
Default is ALLSIDS. System Identifier of system to be deleted.

Listing high-level qualifiers for the Usage Monitor utility

IBM Z Software Asset Management collects large amounts of usage data. The High-level Qualifier Listing for the Usage Monitor utility creates a list of high-level qualifiers for the products that are to be identified.

Following 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.*)
```

The high-level qualifier listing process is automated in the Inquisitor Import job. The high-level qualifier listing is written to a data set, and this data set is concatenated to the UMONIN control file for the Usage Monitor program.

Running the High-level Qualifier Listing for the Usage Monitor utility

To run the High Level Qualifier for the Usage Monitor utility, use the job HSISLLST in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

TPARAM parameters

DSN=
Db2 location. Value assigned, as defined in HSISCUST.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

Updating the TPARAM table

The TPARAM table in the repository can be set to an inconsistent state due to failures in jobs that update the repository tables. You can reset a parameter in the TPARAM table to rectify this inconsistent state.

To run the TPARAM table update, use the job HSISTPRM, in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

SYSIN parameter

```
UPDATE &REPZSCHM.TPARAM SET FVALUE = '0' WHERE FKEY = 'PROCRUN';
```

Tagging unidentified products with the Product Tagging utility

The Product tagging utility allows you to create local entries for your in-house software in the Local Knowledge Database. It is not intended for creating IBM or ISV software, as this may cause unpredictable results. If you need to add products or versions that are not being identified, then notify IBM Support, so that they can then update the GKB. On certain occasions, IBM Support may consider your request to add IBM or ISV software to the LKB database, but contact them first before doing so.

Note: Using options from the Administration tab of the Analyzer is the preferred method for updating and managing the contents of the Local Knowledge Base (LKB). Use of the Analyzer will ensure that the LKB is updated in a logically consistent way. Large numbers of programs can be tagged much more efficiently by the Product Tagging utility than by the Analyzer. You can use HSITAGP to create and update tag data members at any time, but the tag data will only be extracted by the Inquisitor when the OLDTAG keyword is specified in the INQPARM program parameter string that is passed to HSIPINQ.

Product tagging process

Product tagging is a manual process where you must provide the product name, the vendor, and the location of the programs. The Product Tagging utility uses the same method as the Inquisitor program to scan the programs and records the results in dedicated program members.

The SYSIN file contains the control statements that describe which licensed programs are to be tagged. This file contains the program name, vendor name, product identifier, and product version. The program library which contains the software to be tagged is allocated to the SYSLIB file.

You can have only one set of identifying attributes for each program name. If conflicting attributes are found for one or more program names, the Product Tagging utility issues a message and stops.

Information about all discovered programs relating to the nominated product is compiled into a single object module. This module is written to the scanned library allocated to SYSLIB file or to the program library allocated to the optional TAGREDIR file. Using the TAGREDIR file, you can nominate to keep all tag data separate from licensed program software. The TAGREDIR file data sets must be included in the standard Inquisitor scan processing, even if these data sets contain no other program.

The tag data members created by the Product Tagging utility are recognized by the Inquisitor (by their SSI value) during normal program library scanning. The Inquisitor program extracts the tag data from the member contents and writes it to an output file. The Inquisitor import process uses these program tags to maintain entries for the programs in the local knowledge base. The match engine can then accurately identify the tagged product level, regardless of which library the product is deployed to and which system the data is collected from.

Each time you run the Product Tagging utility, it scans a single library and tags a single software product, or optional feature of a product. For products with multiple program libraries, each library is processed in a separate job or step. To ensure effective software identification by the match engine as it processes each library, use the OPTION statement to differentiate the identification entities between the different libraries of a product. Do not tag distribution libraries.

You can override the default output member name of @HSIPTAG by specifying a TAGMEM statement. All output members from the Product Tagging utility are flagged with an SSI value of X'D7E3C1C7', which is 'PTAG' in EBCDIC.

If there is no preexisting member of the same name, the Product Tagging utility creates a new program member to contain the tag data. If a member exists, the new tag data is added to the existing data that relates to other products or optional features. Any data relating to the same software identified by {VENDOR + PRODUCT + OPTION + VERSION} is replaced. 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. These data members are 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 processing with the Product Tagging utility, delete the tag data members which are identified by their SSI value. If you are using ISPF, employ the **SORT SSI** member list command.

The software processed when you run the Product Tagging utility has a key of {VENDOR + PRODUCT + OPTION + VERSION}. If non-key data items, such as the values specified in the PPNUM or LICENSED statements are incorrect, you can correct them by fixing the input statement values and rerunning the utility. This action replaces all non-key tag data. However, if a key data item is incorrect, it will not be erased by running the Product Tagging utility with the correct data.

If you are processing libraries that are not dedicated to a single licensed program, use member name masking to prevent tagging programs not related to that product. Some installations place multiple software products in a combined common library. If the products are tagged before they are combined, you must use different tag data member names.

Product tagging job and control statements

You use the HSISPTAG job in the JCLLIB to run the Product Tagging utility. This job is generated from the HSISCUST post-installation customization job. You input control statements using the SYSIN file.

General syntax rules are:

- 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 non-blanks 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 SELECT, the specified value is deemed to start with the first non-blank after the statement type name.
- Statements can be placed in any order. All statements are processed before any tagging activity commences.
- SELECT is the only statement type which can be supplied more than once in an input file.
- SELECT is the only statement type which can be continued over more than one record.

The following table lists all of the statement types that you can use with the Product Tagging utility:

Table 24. Product Tagging utility statement types

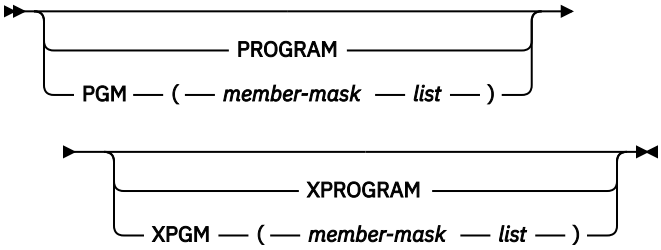
Statement Type	Value	Default Value	Required	Maximum length
VENDOR	Vendor name	-	Yes	30 bytes
PRODUCT	Product name	-	Yes	50 bytes
PPNUM	Licensed program number	blanks	No	16 bytes

Table 24. Product Tagging utility statement types (continued)				
Statement Type	**Value**	**Default Value**	**Required**	**Maximum length**
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
SELECT	Program name filter	PGM(*)	No	8 bytes per mask

SELECT is not a value-oriented statement type. It has operands which have values specified in parentheses. The PROGRAM or PGM inclusion operand can be abbreviated to P. The XPROGRAM or XPGM exclusion operand can be abbreviated to XP.

The Tagger stops parsing a SELECT record and the current statement continues on to the next record whenever a continuation character is encountered. Valid continuation characters are plus and hyphen. A continuation cannot occur within an operand name, or a value mask.

SELECT syntax



member-mask

A string up to 8 bytes in length, representing one or more possible member names of a PDS or PDSE. Use a percent sign to indicate that any single character is to be considered a match in the exact location of the compared character string. Use an asterisk to indicate that any zero or more characters are a match.

Product tagging examples

Three examples are provided to show the ways that you can use the Product Tagging utility to tag unidentified products.

Example 1

A company called ISV has created a build of several programs (build 97) it is developing under the Swisho4U brand. The data sets created by this build have their own disk volume called BLD097. The tag data is to be redirected to a data set dedicated for this purpose.

```
//STEP1   EXEC PGM=HSITAGP
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DSN=S4U.LOADLIB,DISP=SHR,UNIT=3390,VOL=SER=BLD097
//TAGREDIR DD  DSN=S4U.TAGLIB,DISP=SHR
//SYSIN    DD  *
VENDOR   ISV
PRODUCT  Swisho4U
VERSION  BUILD097
/*
```

Example 2

The BigBiz Inc. data center is about to deploy the contractor data processing component for version 4.2 of its internally developed human resources application called HU-MAN. The software is tagged in its own library, but the default tag member name is not used in case it is later loaded into a program library common to several applications. All programs in HU-MAN have names beginning with HU, but the contractor component is the only component which has program names beginning with HUC. The relevant program library can be accessed by using the catalog.

```
//TAGRUN  EXEC PGM=HSITAGP
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DSN=HUMAN.V4R2M0.LOAD,DISP=SHR
//SYSIN    DD  *
VENDOR   BIGBIZ INCORPORATED
PRODUCT  HU-MAN Human Resources Management
OPTION   Contractor Handling
VERSION  04.02.00
TAGMEM   HUMANT@G
SELECT   PGM(HUC*)
/*
```

Example 3

Version 1.5 of the product MVSBLOAT from MiscWare has been deployed on a system which has a dedicated tag data library called SYS2.TAGLIB. Link list programs for the product have been placed in SYS2.LINKLIB and ISPF application modules have been placed in SYS2.ISPLLIB. The product does not have optional features, but only the base component installed. All the installed programs have names beginning with MVSB. The OPTION statement is used to ensure that the contents of each library can be identified by the Match Engine.

```
//STEP1   EXEC PGM=HSITAGP
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DSN=SYS2.LINKLIB,DISP=SHR
//TAGREDIR DD  DSN=SYS2.TAGLIB,DISP=SHR
//SYSIN    DD  *
VENDOR  MiscWare
PRODUCT MVSBLOAT
OPTION  BASE (Batch)
VERSION 01.05.00
TAGMEM  $$OEMTAG
SELECT  PGM(MVSB*)
/*
//STEP2   EXEC PGM=HSITAGP
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DSN=SYS2.ISPLLIB,DISP=SHR
//TAGREDIR DD  DSN=SYS2.TAGLIB,DISP=SHR
//SYSIN    DD  *
VENDOR  MiscWare
PRODUCT MVSBLOAT
OPTION  BASE (Dialogs)
VERSION 01.05.00
TAGMEM  $$OEMTAG
SELECT  PGM(MVSB*)
/*
```

Importing Subcapacity reporting data with the SCRT Import utility

The SCRT (Subcapacity Reporting Tool) Import utility imports data (CSV files) which are generated from the IBM Subcapacity Reporting Tool. The import tool now also supports the importing of Tailored Fit Pricing (TFP) data. New reports to show TFP data are available in the Analyzer under the Asset tab.

MSU vs Container consumption

You need to be aware of the differences between MVM SCRT and TFP SCRT. If you want to see MSU consumption by Product, then you need to import SCRT as MVM. That way you can use the Asset reports to view MSU consumption at the product level. Be certain you choose the Metric SCRT MSU in the relevant Asset reports. If you just want to see MSU consumption at the container level, then import TFP data. TFP data does not have MSU consumption by Product. TFP has its own reports in the Analyzer.

Running the SCRT Import utility

To run the SCRT import utility, use the job HSISSCRT in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

Data input

DDNAME CSVIN contains the CSV output from the IBM ® SCRT tool which can be from a data set with DSORG of PS or PO. Binary uploaded CSV files are supported. DDNAME SIDMAP maps duplicate SMFIDs to a unique SID. The SCRT Import utility handles data where the same SMFID is used on multiple machines concurrently. DDNAME PIDMAP allows users to use SCRT data from IBM products and associate the same usage to an ISV product.

Data input example

Map SMFID on specific machines to your desired SID. As described in this example, when processing data for machine serial 11111, SMFID of IP01, assign a SID value of QIP1, where QIP1 is a SID found in the repository. More than one SMF ID can be entered. Ignore this example if there is no SMFID to map.

```
//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 a SID value found in the repository.

Example for PIDMAP

To allow ISV products to use the same SCRT as an IBM product this DD allows you to map the different PID's. Please make sure that you use the correct ISV and IBM PID's when using this DD.

```
//PIDMAP DD *
5615-DB2=ISV-001
5694-A01=ISV-099
/*
```

IBM PID

The IBM PID in the SCRT that will map to the ISV PID.

ISV PID

ISV PID that will be mapped to from the IBM PID.

Data 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 SCRT tool. This may be a DSORG=PO or PS data set.

TPARAM parameters

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

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

GKBSCHEMA=

Global Knowledge Base qualifier for z/OS. Name of qualifier is *&GKBZSCHM*_GKB7.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

Extracting data with the XML Export utility

The XML Export utility extracts information in XML format that you can then import into IBM Control Desk (ICD).

The extracted information can be either:

- A catalog of the products that are installed in your system.
- A catalog of the products defined in the Global Knowledge Base (GKB).

Running the XML Export utility

To run the XML export utility, use the job HSISKBT, in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

The output XML file generated from this utility needs to be transferred by FTP to a distributed environment and then loaded into IBM Control Desk (ICD). The XML file must be translated from EBCDIC to ASCII.

TPARAM parameters

SSID=
Db2 subsystem name. Value assigned, as defined in job HSISCUST.

SCHEMA=
Repository qualifier or Global Knowledge Base qualifier.

a) Using the Repository qualifier value means that a catalog of products installed on your site is selected.

b) You can also use the Global Knowledge Base qualifier value. This would mean that a catalog of all products defined in the Global Knowledge Base is selected.

Transferring output XML by FTP

The output XML file that is generated when you run the XML Export utility must be transferred by FTP to a distributed environment before you can load it into IBM Control Desk (ICD).

Procedure

1. To connect to the host system, in a command line, enter the following command:

   ```
   C:\temp ftp host name
   ```

2. When prompted, enter your user name and password.
3. To set the input to ASCII format, enter the following command:

   ```
   ftp > quote type a
   ```

4. To transfer non-ASCII characters, enter an ENCODING command before you enter the GET command:

   ```
   quote site ENCODING=MBCS MBDATACONN=(IBM-939,UTF-8)
   ```

   This example specifies encoding for a Japanese code page.

5. To specify the location of the file to transfer, enter the following command:

```
ftp > get 'hsiinst.SWKBT.XML' C:\XML.FILE
```

6. To complete the FTP transfer, enter the following command:

```
ftp > exit
```

Loading data into a staging table (for Db2 only)

Use this utility to load data from a UMON/ZCAT file into the TSTAGE table. Once the TSTAGE table is populated, the data can be browsed using an Analyzer report. Pre-loading into a table that is indexed speeds up the process.

As the search fields are now indexed, there will be better performance when searching for usage data .

Data input

DDNAME INSTAGE contains the UMON or ZCAT file. The file can either be a zipped or unzipped file and multiple UMON/ZCAT files are allowed as input. However, the concatenated UMON/ZCAT files must all be zipped files or all unzipped files.

Note: To access the loaded data, click on the checkbox: 'Use Preloaded data' in the Analyzer report, 'Usage Monitor File Detail' from the Discovery Tab.

Running the loading utility

To run the loading utility, use the job HSISSTAG in the JCLLIB. This job is generated from the HSISCUST post- installation customization job.

TPARAM parameters

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

DSN=
Db2 location. Value assigned, as defined in job HSISCUST.

REPSCHEMA=
Repository qualifier. Name of qualifier is *&REPZSCHM*.

Reporting LMOD ownership from the CSI

The CSI scanning utility scans LMOD ownership in target zones either from the GLOBAL CSI or a TARGET CSI.

Running the CSI Scanner utility

To run the CSI Scanner utility, use the job HSISCSI in the JCLLIB. This job is generated from the HSISCUST post-installation customization job.

Data input
Input for the CSI file is defined in the PARM of the //EXEC.

Data output
The binary output is captured in the DDname HSIXIBIN.

Optional:

JCLOUT
Generated matching Inquisitor JCLs.

PARMOUNT
Generated parameters used in the Inquisitor JCLs.

Compressing and decompressing data sets with the HSIZIP utility

HSIZIP is a utility program that can compress a sequential or partitioned data set into a zip archive, decompress the contents of a zip archive into a sequential or partitioned data set, and report on the contents of a zip archive.

In this context, an archive is a sequential file that contains one or more logical files for the purpose of reducing the space occupied by the data. The archive can serve as a backup and convenient transport format for the data it contains. The Inquisitor and Usage Monitor components usually create zip archives to contain the data that they collect.

The HSIZIP utility has two compress and decompress functions: one for text data, and one for binary data.

Text data processing with the HSIZIP utility

The HSIZIP utility processes text data to be compatible with the zip processing utilities available on other platforms.

When compressing a text record, the HSIZIP utility performs the following processing:

- Translates EBCDIC line feed (LF) characters (x'25') to periods.
- Translates EBCDIC data to ASCII data.
- Appends an ASCII carriage return line feed (CRLF) sequence (x'0D0A') to encode the record extent.
- Compresses the data and writes it to the archive.

Each compressed member is marked as an ASCII text file by setting the internal attribute value in the central file header to 1.

The following input data set attributes are also stored in the zip header extended field:

- Data set organization
- Record format
- Block size
- Logical record length

When decompressing text data, the HSIZIP utility performs the following processing:

- Accumulates data until an ASCII LF (x'0A') is encountered.
- Truncates the trailing ASCII carriage return (CR) (x'0D') if present in accumulated data.
- Translates the ASCII data to EBCDIC and writes the data as a single record.

During compression, records read from data sets with fixed-length records have their trailing blanks truncated before being compressed. After being decompressed, short records to be written to data sets with fixed-length records are extended with blanks to the required length.

The translation tables used for conversion between EBCDIC and ASCII that are originally sourced from the EZAESENU member in the SEZATCPX library are reciprocal, so that applying one translate table and then the other yields the original data. Consequently, all EBCDIC single byte character set (SBCS) and double byte character set (DBCS) text can undergo a ZIP and UNZIP cycle without corruption.

Binary data processing with the HSIZIP utility

The HSIZIP utility processes binary data in order to preserve record boundaries, while other platforms typically consider binary data to be a byte stream without structure.

When compressing a record of binary data, the HSIZIP utility performs the following processing:

- So that record boundaries can be preserved, the following is done depending on the input record format:
  - For fixed-length records, no additional data preparation is done.
  - For variable-length records, the record descriptor word (RDW) is retained as part of the data.
  - When the record format is undefined, each block is prefixed by an RDW where the first two bytes contain the length of the block including the RDW, and the third and fourth bytes contain zeros.
- Compresses the data and writes it to the archive.

Each compressed member is marked as a binary file by setting the internal attribute value in the central file header to 0.

The following input data set attributes are also stored in the zip header extended field:

- Data set organization
- Record format
- Block size
- Logical record length

When decompressing binary data, the HSIZIP utility performs the following processing:

- To establish the length of the record, the following is done depending on the record format of the original input data set:
  - For fixed-length records, the original record length is used.
  - For other record formats, 4 bytes from the archive are decompressed and examined to determine if they form a valid RDW. If so, the RDW length indication is used, and if not, then the data is treated as a byte stream where record boundaries do not need to be preserved.
- Data is decompressed and written as a record of the determined length. Maximum-length records are written when the data is assessed to be a byte stream.

During decompression of binary data, the embedded RDWs are checked for validity. If an RDW does not indicate a positive length greater than 4 or does not end with two bytes of zeros, the HSIZIP utility switches to byte stream mode. In byte stream mode, the utility considers data as a stream of bytes without an inherent record structure. If the RDW that fails the validity test is the first four bytes of the file, the resultant decompression is broadly compatible with the decompression that most other platforms perform and the utility issues an informational message. If the RDW that fails the validity test is not at the start of the file, the utility issues a warning message, sets the final condition code to be greater than zero, but continues processing so that the output data is available for any necessary data recovery activity.

HSIZIP program parameters

The HSIZIP utility can accept up to two program parameters. The first parameter specifies the function the program is to perform and the second parameter can provide a data definition override list for programs that dynamically invoke the utility.

When you invoke the HSIZIP utility as a stand-alone batch program, the PARM value on the EXEC statement specifies the functional request. DD statements define the details of the following files:

- The SYSPRINT report file
- The SYSUT1 input file
- The SYSUT2 output file

You can specify program parameters in the function request in mixed case. The following information describes valid program parameters.

none
If you omit the program parameter, usage notes will be printed to SYSPRINT and LIST processing will be initiated. If SYSUT1 is not allocated the program completion code will be set to 1.

LIST
If you specify this parameter, the utility produces a list of the entries in the central file directory.

TEST
This function will list the local file headers as well as the entries in the central file directory, check that the stored and actual values of some properties match, then perform a trial unzip of each file in the zip archive.

ZIP or ZIP=*filename.ext*
Use this parameter to compress a sequential data set into a new archive with a single constituent file assigned the name specified in the parameter. If no name is specified in the parameter, the name **seq.txt** is used. The data is treated as text.

ZIP or ZIP=.ext***
Use this parameter to compress a partitioned data set into a new archive where each member is loaded as a separate zipped file within the archive. The PDS member name is used to name each corresponding archive member. If an asterisk and a file extension are specified, then the file extension will be appended to the name of the zipped version of each real PDS member. The data is treated as text.

ADD or ADD=*filename.ext*
Use this parameter to compress a sequential data set into an existing archive. A new archive member will be created with the name specified in the parameter. If no name is specified in the parameter, the name **seq.txt** is used. The data is treated as text. There is no dependency on the text or binary nature of files already present in the archive. To retain data access, ensure that all resultant archive member names are unique.

ADD or ADD=.ext***
Use this parameter to compress a partitioned data set into an existing archive where each member is loaded as a separate zipped file within the archive. The PDS member name is used to name each corresponding archive member. If an asterisk and a file extension are specified, then the file extension will be appended to the name of the zipped version of each real PDS member. The data is treated as text. There is no dependency on the text or binary nature of files already present in the archive. To retain data access, ensure that all resultant archive member names are unique.

UNZIP or UNZIP=filenamemask
Use this parameter to decompress an archive into a partitioned data set and load each zipped file into a separate member. The parameter restores data sets from archives made by the HSIZIP utility with **PARM=ZIP**. If the output data set is sequential, only the first file in the archive is unzipped. You can use the file name mask specification to filter the files to be unzipped.

ZIPBIN or ZIPBIN=*filename.ext*
Use this parameter to compress a sequential data set into a new archive with a single constituent file assigned the name specified in the parameter. If no name is specified in the parameter, the name **seq.bin** is used. The data is treated as binary and no translation is performed.

ZIPBIN or ZIPBIN=.ext***
Use this parameter to compress a partitioned data set into a new archive where each member is loaded as a separate zipped file within the archive. The PDS member name is used to name each corresponding archive member. If an asterisk and a file extension are specified, then the file extension will be appended to the name of the zipped version of each real PDS member. The data is treated as binary and no translation is performed. There is no dependency on the text or binary nature of files already present in the archive.

ADDBIN or ADDBIN=*filename.ext*
Use this parameter to compress a sequential data set into an existing archive. A new archive member will be created with the name specified in the parameter. If no name is specified in the parameter, the name seq.bin is used. The data is treated as binary and no translation is performed. There is no dependency on the text or binary nature of files already present in the archive. To retain data access, ensure that all resultant archive member names are unique.

ADDBIN or ADDBIN=.ext***
Use this parameter to compress a partitioned data set into an existing archive where each member is loaded as a separate zipped file within the archive. The PDS member name is used to name each

corresponding archive member. If an asterisk and a file extension are specified, then the file extension will be appended to the name of the zipped version of each real PDS member. The data is treated as binary and no translation is performed. There is no dependency on the text or binary nature of files already present in the archive. To retain data access, ensure that all resultant archive member names are unique.

UNZIPBIN or UNZIPBIN=filenamemask
Use this parameter to decompress an archive into a partitioned data set and load each zipped file into a separate member. The parameter restores data sets from archives made by the HSIZIP utility with **PARM=ZIPBIN**. If the output data set is sequential, only the first file in the archive is unzipped. Use the file name mask specification to filter the files to be unzipped.

The filenames and filename masks that you specify in program parameters must not exceed 128 bytes in length. File name mask matching is case insensitive. The following characters are generic masking characters for filename masks:

- ? (question mark) matches any single character.
- * (asterisk) matches any zero or more contiguous characters.

If the function request is absent or invalid, the utility writes usage notes to the report file. If the request is absent, the utility attempts to run the LIST function.

Controlling zip compaction

For the program parameter values described above, wherever the parameter begins with the characters **ZIP** or **ADD**, the values **ZP*n*** or **ADn** can be substituted respectively, where **n** is a decimal digit in the 0 to 9 range which specifies the compaction that the zip process is to use.

A value of 0 specifies that the ***shrink*** method is to be used, which is the method the HSIZIP program always used in releases earlier than 8.2.

A value in the 1 to 9 range specifies the corresponding compaction level of the ***deflate*** method. As the compaction level number increases, so does both the data compression and the CPU time consumed by the zip process.

The default is the fastest ***deflate*** compaction level. That is, **PARM=ZIP** is equivalent to **PARM=ZP1** and **PARM=ADD** is equivalent to **PARM=AD1**.

Handling EBCDIC Code Page 1047 Text

When EBCDIC text is being zipped for the simple purpose of reducing its volume, the precise translation into ASCII is not important as along as it is exactly reversed before the unzipped EBCDIC text is processed. However, when the EBCDIC text is to be transported to a platform using a character set based on ASCII, various code page translation issues may arise. A frequent issue is the differences between EBCDIC code pages 037 and 1047, the latter being the default code page of z/OS UNIX.

By default, HSIZIP uses translate tables copied from SEZATCPX library member EZAESENU which caters for code page 1140 which is equivalent to code page 037 with a Euro currency sign modification. HSIZIP can also be directed to use the translate tables copied from SEZATCPX library member EZAAS010 which cater for code page 1047.

To use the relevant alternate code page 1047 translate table, a character in the program parameter which specifies the request is changed to an A.

For compression requests, the second character is changed to an A. So, parameter values of AAD, AA0, ZAP, and ZA1 are equivalent to alternate translation requests for ADD, AD0, ZIP and ZP1 respectively. Any of the ZIP and ADD requests can have the second character changed to A for this purpose, although the alternate translate specification will be ignored for binary data requests.

For decompression requests, change UNZIP to UNZAP to use the alternate translation from ASCII to EBCDIC.

There is no metadata present in a zip archive created or modified by HSIZIP to indicate which translate table was used to translate text data. Be careful to use consistent translations for compression and decompression when consuming zipped data on EBCDIC-based platforms such as z/OS.

HSIZIP files

The HSIZIP utility uses the following files:

- SYSPRINT is a report file. RECFM=VBA and LRECL=137 are used in the DCB.
- SYSUT1 is an input file that describes the data set that contains data to be zipped or the zip archive that contains data to be listed or unzipped.
- SYSUT2 is an output file that contains the results of a compression or a decompression operation. This file is not required by the LIST and TEST functions.

The HSIZIP utility does not support spanned records for any file. The main compression and decompression input and output to archive files uses the queued sequential access method (QSAM) locate mode. Apart from the lack of support for spanned records, an input archive allocated to SYSUT1 can have any valid record format and reside on any device that can be read by QSAM. An archive allocated to SYSUT2 must have variable-length records and support update-in-place processing. In effect, a SYSUT2 file must be an MVS DASD data set that is not also a compressible extended-format data set.

Dynamic invocation of the HSIZIP program by other programs

Other programs can call the HSIZIP utility to perform compression and decompression processing requests. When the HSIZIP utility receives control, it examines the program parameter list and proceeds accordingly.

The first program parameter must begin with a halfword counter indicating the length of the function request text that immediately follows. The format is the same format as the system uses to pass the parameter specified in the PARM operand of the EXEC statement in JCL.

A second program parameter can be specified to override the default file names used by the HSIZIP utility. If the value of the halfword length indicator at the start of the parameter is not a multiple of 8 or is not less than 256, the HSIZIP utility ignores it. A series of 8-byte file name entries immediately follow the length indicator and each can specify the DD name to use instead of the default name. Set a slot to 8 bytes of zeros to avoid overriding that particular default file name. SYSPRINT, SYSUT1 and SYSUT2 correspond to the sixth, eighth and ninth file name slots respectively.

HSIZIP data set support

The data control block (DCB) attributes of the original data set that the HSIZIP utility compresses are stored in zip header fields where possible, but the information is not used to supply output data set attributes during unzip processing. The success of a compress and decompress cycle requires the user to supply suitable DCB attributes for the ultimate destination of the data.

The following points are provided to help you to assess whether the HSIZIP utility can successfully process a data set:

- When processing a whole partitioned data set, the file name specified after **ZIP=** or **ADD=** is ignored, unless it begins with an asterisk followed by a period, when it becomes a file extension specification.
- When ZIP processing detects that a PDS member is a zip archive or a GIF file, the member is stored as a byte stream as is without attempting further compression or record boundary preservation.
- ZIPBIN processing of PDS members containing zip archives usually causes the compressed size to be larger than the uncompressed size, due to the inability to further compact the data and the insertion of RDWs to preserve record boundaries. So, if the only non-text data in a PDS is in members which are themselves zip archives, specify ZIP rather than ZIPBIN to minimize the resultant file size.
- When using ADD or ADDBIN, avoid duplicate file names in the resultant archive.

- You can use the ADD and ADDBIN parameters to create an archive with a mixture of text and binary file members.
- The binary or text nature of an unzip process is set by the program parameter and not from the attribute values in the file header.
- When the HSIZIP utility creates a zip archive, the data set name of the input file is stored as the zip archive comment.
- PDS member user data such as system status information (SSI), ISPF statistics, and load module attributes are stored in the comment field of the central file header of the archive member and can be restored during unzip operations.
- Alias members are stored as files with zero bytes. The alias member data is preserved only if the real member associated with the alias member is also processed.
- Use ZIPBIN and UNZIPBIN when processing load module libraries.
- The HSIZIP utility cannot restore program PDSE data sets because only the program binder can write to program PDSEs. There is no restriction on data PDSEs.

HSIZIP return codes

When you run the HSIZIP utility, several codes are returned that indicate whether the program ran successfully.

Table 25. HSIZIP utility return codes

Return code	Description
0	Request processed successfully.
1	Usage notes were printed and the SYSUT1 DD was not allocated.
2	Warning message issued. The warning is for a condition that does not affect the operation of the current request, but will probably impact on the intended use of the file created by the request.
4	No data was found to process, or an I/O error was encountered.
8	An error occurred. Look at SYSPRINT for more details.
Other	As set by another routine. Look at SYSPRINT for more details.

Browsing zipped data

Components such as the Inquisitor programs, the Usage Monitor, and the ZCAT utility deal with sequential data sets containing zip archives. Processing the data with a zip algorithm allows the same information to be conveyed in a fraction of the original size, but it does mean that browsing the file to quickly confirm the nature of its content becomes a more involved process.

One advantage of a file browser over an editor is that the browser can quickly present the data from the start of a file for inspection without reading in the whole data set, whereas editors typically read the complete data contents before proceeding. Similarly, a data set containing zipped data could be unzipped with the result being browsed in the usual way, but this means that space for all the unzipped data is required which tends to defeat the purpose of compressing the data.

Browsing zipped data using HSIBRZIP

The HSIBRZIP program can be used to browse the unzipped contents of a data set without requiring the unzipping of the entire data set. A REXX EXEC called HSIIBRWZ is shipped in the SHSIEXEC library to provide a convenient interface to invoke the HSIBRZIP program.

To make the facility available for use, customize the HSIIBRWZ EXEC and place it in a suitable library giving it a suitable member name. (When installing it into your local library, you can call it HSIIBRWZ, or you can give it a simpler name, such as BRZIP or whatever you find suitable.) The customization process consists of supplying the data set name of the program library containing the HSIBRZIP program.

HSIBRZIP invokes the BRIF service of ISPF, and so it requires an ISPF environment for execution. The HSIBRZIP EXEC expects an operand of a data set name, and so is suitable for general use under ISPF including as a line command in a data set list created by option 3.4.

HSIBRZIP only unzips enough data to be able to provide the records selected by the user for browsing. For example, if there are ten million records but the user only scrolls down to view the first hundred records in the browse session, then only 100 records need to be unzipped. Unzipped records are staged in a data space so that scroll up requests can be satisfied by providing records from the data space without the need to interrupt the current progress of the suspended unzip process. The unzip process is resumed when previously unread records need to be accessed.

HSIBRZIP has several limitations and behavioral characteristics:

- All zipped data is assumed to be ASCII text, and is translated to EBCDIC before display.
- The maximum record length without wrapping on to a new line is 1024 bytes.
- The name passed to BRIF to display as the file name is the name of the first or only file in the zip archive.
- If the end of a file is reached, before showing data from the next file, HSIBRZIP will insert a record containing the following message:

  ```
  { HSIBRZIP reached end of file  -  Start of file newname }
  ```

  where ***newname*** is the name of the next file being unzipped from the same archive.
- If HSIBRZIP recognizes records as having come from the Inquisitor or the Usage Monitor then it will insert records into the browse data to provide column headings for data items within the recognized records. Such inserted lines will be repeated whenever the record type is different from the previous record, and will have the following form:

  ```
  { detailsThis line was inserted for display by HSIBRZIP }
  ```

  where ***details*** describes items present in the subsequent record(s).
- HSIBRZIP cannot present data that would cause it to read more records than can be stored in the data space, either because local limits failed a data space extend request, or because of the 2 gigabyte size limit of data spaces.

Browsing active collection data

The Usage Monitor collects program usage information and writes out the collected data at least once each day. Once the data is in a data set, it is possible to browse it to determine if some expected usage was collected or not. Sometimes it would be advantageous to be able to determine if expected data collection has taken place without waiting for the collection cycle to end, either at the expected time or because it was triggered manually to facilitate data access.

Browsing active collection data using HSIZPEEK

The HSIZPEEK program provides a way of accessing data in the active collection repository. It writes a report to the UMONDATA DD which can be directed to SYSOUT or to a sequential data set. The UMONDATA file has fixed-length records with an LRECL set by the HSIZPEEK program. The record length depends on the items being reported and so is subject to change, but is currently 289 bytes. HSIZPEEK can be run in a batch job step or in a TSO session.

The HSIZPEEK REXX EXEC is shipped in the SHSIEXEC library and when executed presents the data from HSIZPEEK in an ISPF Browse session. Before using HSIIPEEK customize it by supplying the data set name of the program library containing the HSIZPEEK program, and store the EXEC with a suitable member name in your local REXX EXEC library.

The contents of the UMONDATA file are not a documented programming interface. However, the first section of the report contains one line for each detail usage record that will be written by the writer task at the end of the collection cycle, and so a simple FIND command in an HSIIPEEK ISPF Browse session can

tell you whether usage for a specific program has been detected in the current collection cycle so far or not. You can find program library data set name and job name details on the same report line.

IBM Technical Support may ask for information from HSIIPEEK reports while investigating and diagnosing problems.

Verifying database changes since the product was released

This utility verifies database changes that were introduced after the product was released.

To run the verification, use job HSISIVPD in the JCLLIB. This job is generated from the HSISCUST post-installation job.

Configuring language support

IBM Z Software Asset Management includes Japanese language support for MVS Message Service (MMS) message information. You can also configure the Analyzer utility to create and view reports in Japanese.

Configuring Japanese messages

To configure messages in the Japanese language, there are two areas to customize. You must compile the Japanese language MMS messages, and then you must enable messages in the Japanese language in the Language Environment®.

About this task

Japanese language MMS message information is contained in the hsi.SHSIMJPN program directory. You must compile the message file and then install the most recent system runtime message file. The HSISMCMP job in the JCLLIB library compiles MMS messages. This job is generated by the HSISCUST post-installation customization job.

Procedure

1. Run the HSISMCMP job to compile the MMS files into a system runtime message file, for example the `hsi.MMSJPN99` file.

   The HSISMCMP job is generated by the HSISCUST post-installation customization task.
2. Create an entry named **MMSLSTJ9** in the z/OS PARMLIB library, with the following values:

   ```
   DEFAULTS LANGCODE(JPN)
   LANGUAGE LANGCODE(JPN) DSN(SYS2.MMSJPN99) CONFIG(CNLJPN00)
   ```
3. Enter the following MVS system command to install the system runtime message file:

   ```
   SET MMS=J9
   ```
4. Enable Japanese messages for the Inquisitor Import and the Usage Import, Summary, and Deletion components.

   To configure the language environment, refer to the following documents:

   - For information about the Language Environment `MSGFILE` and `NATLANG` options, refer to the *Language Environment Programming Reference (SA227562).*
   - For information about specifying Language Environment runtime options, refer to the *Language Environment Programming Guide (SA227561)*.
   - For information about setting `NATLANG(JPN)` as an installation default, refer to the *Language Environment Customization (SA22-7564).*

Enabling the Analyzer utility for Japanese

You can configure the Analyzer utility so that you can view and create reports in Japanese.

Procedure

1. In the Analyzer HSIJANLO started task, update the DD statements with the following commands:

```
//HSICUST hsi.SHSIPARM(HSISANCJ)
//HSINLS hsi.SHSIANL1(HSINLSJP)
```

2. Perform the following tasks to customize your Japanese Analyzer reports:
   a) Copy the hsi.SHSIPARM(HSISANCJ) parameter to the hsiinst.PARMLIB(HSISANCJ) library.
   b) Modify the hsiinst.PARMLIB(HSISANCJ) library to customize your reports.
   c) Enter the following command to update the Analyzer HSIJANLO started task:

```
//HSICUST hsiinst.PARMLIB(HSISANCJ)
```

Configuring the Japanese Db2 subsystem for use with IBM Z Software Asset Management

About this task

IBM Z Software Asset Management is implemented with a Japanese Db2 subsystem that is configured with the MCCSID=939 code page (Japanese extended English).

If your Db2 for z/OS system is configured with MCCSID=930, MCCSID=1390, or MCCSID=5026 code page (Japanese extended Katakana), additional customization is required to set up IBM Z Software Asset Management.

For Katakana codepage, data for lower case characters (a,b,c,d,...,z) are stored differently in the databases. For example:

- c = x'83' for Latin-based codepage
- c = x'64' for Katakana codepage

In Katakana, the hex definitions for lower case characters are different from Latin-based codepages.

In IBM Z Software Asset Management, some columns have data stored in lower case. As a result, there are extra considerations in managing the conversion of data for lower case characters. In addition, module names in the GKB Knowledge Base for z/OS (not GKB Knowledge Base for z/OS UNIX System Services), are in encoded format – meaning some of the encoded characters are in lower case characters.

For consistency in IBM Z Software Asset Management, if the Db2 subsystem is defined in the Katakana code page, then all lower case characters must be stored in the Katakana format (value of 'c' must be stored as x'64').

When setting up a Db2 subsystem to support EBCDIC double byte CCSID (coded character set identifiers), in the Db2 for z/OS installation job (DSNTIJUZ), parameter MIXED is set to 'YES' and parameter MCCSID defined with different values (default value is a 65534). Here is an example from DSNTIJUZ:

DSNHDECM CHARSET=KATAKANA

- ASCCSID=1041
- AMCCSID=942
- SCCSID=290
- MCCSID=930 (1390 or 5026)
- GCCSID=300
- USCCSID=367

- UMCCSID=1208
- UGCCSID=1200
- ENSCHEME=EBCDIC
- APPENSCH=EBCDIC
- ENSCHEME=EBCDIC
- APPENSCH=EBCDIC
- DATE=ISO
- DATELEN=0
- DECARTH=DEC15
- DECIMAL=PERIOD
- DEF_DECFLOAT_ROUND_MODE=ROUND_HALF_EVEN
- DEFLANG=IBMCOB
- DELIM=DEFAULT
- IMPLICIT_TIMEZONE=CURRENT
- MIXED=YES

You must perform additional customization in IBM Z Software Asset Management.

Customization are as follows:

Procedure

1. Update the ODBC Initialization file.

   The ODBC initialization file can be found in members:

   PARMLIB (HSISCLI) and PARMLIB (HSISCLIT).

   Do the following in members HSISCLI and HSISCLIT:

   a) Uncomment parameter CURRENTAPPENSCH in the "COMMON” section.
      Here is an example on updating the two values:

      ```
      ; COMMON section
      CURRENTAPPENSCH=939
      ;
      ```

2. Update the following PARMLIB members to include the "CCSID (1027,939,300)" entry for all "LOAD DATA" statements:

   HSISSQ06

   HSISSQ12

   HSISSQ29

   For example in PARMLIB (HSISSQ29):

   ```
   LOAD DATA INDDN TPARAM   LOG YES REPLACE
              CCSID (1027,939,300)
              INTO TABLE TFGKBZ9_IQF7.TPARAM (
              FKEY               POSITION(  1: 64) CHAR(64),
              FVALUE             POSITION( 65:318) CHAR(254))

              LOAD DATA INDDN TCOMPILE  LOG YES REPLACE
              CCSID (1027,939,300)
              INTO TABLE TFGKBZ9_IQF7.TCOMPILERS (
              FCOMPID            POSITION(  3: 17) CHAR(15),
              FCOMPNAME          POSITION( 18: 47) CHAR(30))

              LOAD DATA INDDN TIQFILTR  LOG YES REPLACE
              CCSID (1027,939,300)
              INTO TABLE TFGKBZ9_IQF7.TIQFILTERS (
              FOWNER             POSITION(  3:  3) CHAR(1),
   ```

```
            FIQFILTER          POSITION(  4: 58) CHAR(55),
            FLPARNAME          POSITION( 59: 62) CHAR(4))

            LOAD DATA INDDN TXPCMODU  LOG YES REPLACE
            CCSID (1027,939,300)
            INTO TABLE TFGKBZ9_IQF7.TXPCMODULES (
            FOWNER             POSITION(  2:  2) CHAR(1),
            FMODNAME           POSITION(  3:256) CHAR(254))

            LOAD DATA INDDN TXPCSPEC  LOG YES REPLACE
            CCSID (1027,939,300)
            INTO TABLE TFGKBZ9_IQF7.TXPCSPEC (
            ID                 POSITION(  2:  5) INTEGER,
            FLIST              POSITION(  6: 55) CHAR(50))

            LOAD DATA INDDN TXVENDOR  LOG YES REPLACE
            CCSID (1027,939,300)
            INTO TABLE TFGKBZ9_IQF7.TXVENDORS (
            FOLDVENDORNAME     POSITION(  2: 65) CHAR(64),
            FNEWVENDORNAME     POSITION( 66: 95) CHAR(30))
```

3. Run HSISGKBL to load GKB data with these new LOAD defenitions.
4. Bind the Db2 DSNREXX plan.

   Following are the implementation steps:

   a) Create a copy of db2.SDSNSAMP(DSNTIJTM). This copy should just contain the DSNREXX job step.

   b) Update the job by replacing parameter ENCODING (EBCDIC) with ENCODING (939). See example below:

   c) Run the modified DSNTIJTM job. DSNREXX plan should now have ENCODING (939).

   Example:

```
DSN SYSTEM(DSN)
            BIND PACKAGE(DSNREXX) MEMBER(DSNREXX)  ACTION(REPLACE)
            ISOLATION(CS)-
            LIBRARY('DB2V12.SDSNDBRM')-
            VALIDATE(BIND) CURRENTDATA(NO)-
            ENCODING(939)
            BIND PACKAGE(DSNREXUR) MEMBER(DSNREXX) ACTION(REPLACE)
            ISOLATION(UR)-
            LIBRARY('DB2V12.SDSNDBRM')-
            VALIDATE(BIND)-
            ENCODING(939)
            BIND PACKAGE(DSNREXCS) MEMBER(DSNREXX) ACTION(REPLACE)
            ISOLATION(CS)-
            LIBRARY('DB2V12.SDSNDBRM')-
            VALIDATE(BIND) CURRENTDATA(NO)-
            ENCODING(939)
            BIND PACKAGE(DSNREXRS) MEMBER(DSNREXX) ACTION(REPLACE)
            ISOLATION(RS)-
            LIBRARY('DB2V12.SDSNDBRM')-
            VALIDATE(BIND)-
            ENCODING(939)
            BIND PACKAGE(DSNREXRR) MEMBER(DSNREXX) ACTION(REPLACE)
            ISOLATION(RR)-
            LIBRARY('DB2V12.SDSNDBRM')-
            VALIDATE(BIND)-
            ENCODING(939)
            BIND PLAN(DSNREXX) -
            PKLIST(*.DSNREXX.DSNREXX, -
            *.DSNREXUR.DSNREXX, -
            *.DSNREXCS.DSNREXX, -
            *.DSNREXRS.DSNREXX, -
            *.DSNREXRR.DSNREXX) -
            ACT(REP) ISO(CS) CURRENTDATA(YES) SQLRULES(DB2) -
            ENCODING(939)
            RUN PROGRAM(DSNTIAD)  PLAN(DSNTIA12) -
            LIB('DB2V12.RUNLIB.LOAD')
```

Configuring the Turkish Db2 subsystem for use with IBM Z Software Asset Management

If Db2 for z/OS is configured with SCCSID=1026, additional customization is required in setting up IBM Z Software Asset Management.

About this task

Procedure

1. Update the ODBC initialization file.

   The ODBC initialization file can be found in members:

   PARMLIB (HSISCLI)and PARMLIB (HSISCLIT)

   In the COMMON section, add parameter CURRENTAPPENSCH=37 for both members - HSISCLI and HSISCLIT

   ```
   ; COMMON section
   CURRENTAPPENSCH=37
   ;
   ```

2. Update the following PARMLIB members to include the "CCSID (37)" entry for **ALL** "LOAD DATA" statements:

   (HSISSQ06)

   (HSISSQ12)

   (HSISSQ29)

   Here is an example for (HSISSQ06):

   ```
   LOAD DATA INDDN TPARAM LOG NO NOCOPYPEND RESUME
   NO REPLACE
   CCSID (37)
   INTO TABLE &GKZSCHM_GKB&.TPARAM(
   ```

3. Run HSISGKBL to load GKB data with these new LOAD definitions.

Reference information for IBM Z Software Asset Management

Reference information includes messages, repository table layouts, and performance and tuning.

Repository table layouts

This topic describes the tables in the Repository including column names, types, and length.

Table 26. 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

Table 26. NODE (continued)

Column name	Column type	Column length	Description
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	Integer		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
HW_ID	Smallint		NODE_KEY identifier

Table 27. 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
MODEL_CAPACITY	Char	4	CPU model for capacity planning purpose

Table 28. PRODUCT

Column name	Column type	Column length	Description
SW_KEY	Char	32	Global Unique ID (GUID) for this entry. For SW_TYPE=VERSION this will be the same value as VERSION_GUID For SW_TYPE=FEATURE this will 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	80	Product name, which is a normalized form of Version Name in order to group different versions of products under the same product name

Table 28. PRODUCT (continued)

Column name	Column type	Column length	Description
VERSION	Integer		Version
VERSION_NAME	Char	80	Product Version Title
FEATURE_NAME	Char	80	Product Feature Title
PID	Char	16	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
ALT_PRODUCT_NAME	Char	80	Alternate product name
FOBSERVEDELETED	Timestamp		Date and time that the library was deleted from Inquisitor data.
BUNDLE_NAME	Char	50	Product suite
FTYPE	Char	1	P=Product, S=Suite, X=Exclude, Null
FSUITE_ID	Smallint		ID of the Suite
ALT_VENDOR_NAME	Char	50	Alternate vendor name
OPSYSEVENT_CNT	Char	1	Operating system. Value Z or U

Table 29. PRODUCT_INSTALL

Column name	Column type	Column length	Description
SW_KEY	Char	32	Product GUID
SYSTEM_KEY	Char	32	System GUID

Table 29. PRODUCT_INSTALL (continued)			
Column name	**Column type**	**Column length**	**Description**
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 30. 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 31. 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

Table 31. PRODUCT_USE (continued)			
Column name	**Column type**	**Column length**	**Description**
END_DATE	Date		Date within this Period that usage was for last recorded
LAST_UPDATE_TIME	Timestamp		Time stamp entry was last updated
PRODUCTUSE	Float		Counter field for Product usage

Table 32. 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
JOBNAME	Char	8	Job Name
ACCOUNTCODE	Char	20	First 20 chars of the Job Account Code
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
PRODUCTUSE	Float		Counter field for Product usage

Table 33. 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
IPADDR	Varchar	45	Host TCP/IP address
HOSTNAME	Varchar	256	Host TCP/IP name
FOBSERVEDELETED	Timestamp		Date and time that system was deleted from Inquisitor data

Table 33. SYSTEM (continued)

Column name	Column type	Column length	Description
FOSVERSION	Char	8	Version of the z/OS

Table 34. SYSTEM_NODE

Column name	Column type	Column length	Description
SYSTEM_KEY	Char	32	System GUID
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 35. TACCOUNT

Column name	Column type	Column length	Description
FACCOUNTID	Integer		Account ID
FACCOUNTCODE	Char	20	Job Account Code, truncated to 20 characters

Table 36. TALTERNATE

Column name	Column type	Column length	Description
PRODUCT_NAME	Char	80	Product name, which is a normalized form of Version Name in order to group different versions of products under the same product name
ALT_PRODUCT_NAME	Char	80	Alternate product name
BUNDLE_NAME	Char	50	Product suite

Table 37. TANNOTATE

Column name	Column type	Column length	Description
FTYPE	Char	1	Annotation type: • A – Asset • D- Discovery • B - Both
PRODUCT_GUID	Char	32	Globally Unique ID for VENDOR_NAME + PRODUCT_NAME
FANNOTATION	Varchar	255	Text that is annotation

Table 38. TCHANNEL_PATH

Column name	Column type	Column length	Description
SID	Char	4	Product system ID. By default this is the SMFID
CP_CHPID	Char	4	Channel path identifier
CP_TYPE	Char	5	Channel path type acronym
PERIOD	Date		Month for this entry
CP_DESC	Char	32	Channel path description

Table 39. TCONTROL_UNIT

Column name	Column type	Column length	Description
SID	Char	4	Product system ID. By default this is the SMFID
PERIOD	Date		Month for this entry
CU_DEV1	Char	4	First device number
CU_DEV2	Char	4	Last device number
CU_DEVCT	Integer		Device count for CU record
CU_UNTNM	Char	8	Generic device unit name
CU_CHN1	Char	4	Channel path 1
CU_CHN2	Char	4	Channel path 2
CU_CHN3	Char	4	Channel path 3
CU_CHN4	Char	4	Channel path 4
CU_CHN5	Char	4	Channel path 5
CU_CHN6	Char	4	Channel path 6
CU_CHN7	Char	4	Channel path 7
CU_CHN8	Char	4	Channel path 8
CU_TYPE	Char	6	Control unit type
CU_MODEL	Char	3	Control unit model
CU_MFR	Char	3	Control unit manufacturer
CU_PLANT	Char	2	Control unit plant of manufacture
CU_SEQNUM	Char	12	Control unit sequence number

Table 40. TCSECT

Column name	Column type	Column length	Description
FMODID	Integer		Module ID
FCSECTNAME	Char	40	Name of CSECT

Table 40. TCSECT (continued)

Column name	Column type	Column length	Description
FCSECTSIZE	Char	8	CSECT size
FCOMPDATE	Char	8	Compile date of CSECT
FCOMPTIME	Char	6	Compile time of CSECT
FCOMPNAME	Char	50	Compiler name
FCOMPVERSION	Char	8	Compiler version
FLANGENV	Char	2	LE language type
FARCHLEVEL	Char	3	Architecture level
FOPTIMIZE	Char	1	Optimize level
FDSTATEMENTS	Char	6	Number of statements the DATA DIVISION of a COBOL program
FPSTATEMENTS	Char	6	Number of statements the PROCEDURE DIVISION of a COBOL program

Table 41. TISVEOS

Column name	Column type	Column length	Description
FPOVID	Integer		Product ID
FEOSDATE	Timestamp		End of Service Date

Table 42. TJOBDATA

Column name	Column type	Column length	Description
FJOBNAME	Char	8	Job Name
FJOBID	Integer		Job ID
FJOBTYPE	Char	6	Job Type

Table 43. 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	8	Volser library resides on

Table 43. TLIBRARY (continued)

Column name	Column type	Column length	Description
FTRACKSALLOC	Char	10	Number of allocated tracks
FTRACKSUSED	Char	10	Number of used tracks
FORIGIN	Char	1	Blank - PDS, E - PDSE, V - VTOC, Q - PDS (subtask QSAM scan)
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
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
FCHECKSUM	Char	40	A checksum of module names and sizes in a given library used to determine whether a library has changed
FSTORAGEGROUP	Char	8	The storage group the library belongs to
FLIBRARYCLASS	Char	30	Name of the Library Class

Table 44. TLIBSYS

Column name	Column type	Column length	Description
FLIBID	Integer		Library ID
FLPARID	Integer		LPAR ID
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
FAPFAUTH	Char	1	APFAUTH flag Y or N
FAPFLASTUPDATE	Timestamp		Date and time field was last updated
FLINKLIST	Char	1	LINLIST indicator. If Y, then it is the linklist. If N, then not in the linklist

Table 45. TLOGIQ

Column name	Column type	Column length	Description
FIQDATE	Timestamp		Inquisitor date
FIMPORTDATE	Timestamp		Inquisitor Import date
FSID	Char	4	Product system ID. By default this is the SMFID
FSYSPLEXID	Char	8	z/OS Sysplex name
FFULLREMATCH	Char	1	Y or N. Described in job HSISIQIM
FPRODUCTONLY	Char	1	Y or N. Described in job HSISIQIM
FOSTYPE	Char	4	z/OS or z/OS UNIX System Services
FPLX	Char	1	PLX option (Y or N) as defined in Inquisitor scan
FVERSIONGKB	Char	15	The version of the GKB that the IQ is matched with
FSPNUM	Char	8	APAR number of IQ scan (e.g. OA6nnn)
FFMID	Char	8	FMID of the z/OS
FOSNUM	Char	8	Product ID of the z/OS
FOSVERSION	Char	8	Version of the z/OS
HW_TYPE	Char	4	System z hardware type
HW_MODEL	Char	4	System z hardware model
HW_SERIAL	Char	12	System z hardware serial
FFILTER	Varchar	256	List of filters in the Inquisitor file
FAPARLVL	Char	10	APAR number
FFZVSAMLVL	Char	5	FZVSAM level (e.g. 910)

Table 46. TLOGUI

Column name	Column type	Column length	Description
FUMONDATE	Timestamp		Usage Monitor date
FIMPORTDATE	Timestamp		Usage Import date
FSID	Char	4	Product system ID. By default this is the SMFID
FSYSPLEXID	Char	8	z/OS Sysplex name
HW_TYPE	Char	4	System z Hardware Type
HW_MODEL	Char	4	System z Hardware Model
HW_SERIAL	Char	12	System z Hardware Serial
FFILTER	Varchar	256	List of filters in the UMON file
FAPARLVL	Char	10	APAR number of Usage Import
FFZVSAMLVL	Char	5	FZVSAM level (e.g. 910)

Table 46. TLOGUI (continued)			
Column name	**Column type**	**Column length**	**Description**
FUMAPARLVL	Char	10	APAR number of UMON

Table 47. 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
FSYSPLEXID	Char	8	Sysplex name if in a Sysplex
FMIPS	Integer		Number of MIPS for LPAR
FHW_NAME	Char	10	Configured hardware name
FIPADDR	Varchar	45	Host TCP/IP address
FHOSTNAME	Varchar	256	Host TCP/IP name
FOBSERVEDELETED	Timestamp		Date and time that LPAR was deleted from Inquisitor data
FMSU	Integer		MSU value for the LPAR
FHWMODEL	Char	12	Hardware Model
FSWCAP	Char	12	Software capacity model
FMSU	Integer		MSU value for the LPAR
FHWMODEL	Char	12	Hardware Model
FSWCAP	Char	12	Software capacity model

Table 48. TMACHINE_RESOURCE			
Column name	**Column type**	**Column length**	**Description**
HW_NODE_KEY	Char	32	NODE GUID for Hardware NODE that this System was last running on in this month
LPAR_NAME	Char	10	Logical partition name
PERIOD	Date		Month for this entry
LPAR_NUMBER	Integer		Logical partition number
LPC_NAME	Char	10	LPC name
OSTYPE	Char	8	OS name
ENGINE_CNT1	Char	4	CPU available count

Table 48. TMACHINE_RESOURCE (continued)			
Column name	**Column type**	**Column length**	**Description**
ENGINE_CNT2	Char	4	CPU online count
ENGINE_CNT3	Char	4	CPU dedicated count
ENGINE_CNT4	Char	4	zAAP available count
ENGINE_CNT5	Char	4	zAAP online count
ENGINE_CNT6	Char	4	zAAP dedicated count
ENGINE_CNT7	Char	4	IFL available count
ENGINE_CNT8	Char	4	IFL online count
ENGINE_CNT9	Char	4	IFL dedicated count
ENGINE_CNT10	Char	4	ICF available count
ENGINE_CNT11	Char	4	ICF online count
ENGINE_CNT12	Char	4	ICF dedicated count
ENGINE_CNT13	Char	4	zIIP available count
ENGINE_CNT14	Char	4	zIIP online count
ENGINE_CNT15	Char	4	zIIP dedicated count
ENGINE_CNT16	Char	4	Other available count
ENGINE_CNT17	Char	4	Other online count
ENGINE_CNT18	Char	4	Other dedicated count

Table 49. 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
FOBSERVEDELETED	Timestamp		Date and time that module was deleted from Inquisitor data

Table 49. TMODULE (continued)

Column name	Column type	Column length	Description
LINKEDITDATE	Date		Link edit date of module
FCOMPNAME	Char	50	Name of the Compiler, for example, Enterprise COBOL
FCOMPVERSION	Char	50	Version of the Complier
FCOMPDATE	Date		Compiled date of module
FAMODE	Char	3	Specifies the addressing mode in which the program is designed to receive control.
FRMODE	Char	3	Indicates where the program can reside in virtual storage.
FREUSE	Char	4	The REUS option allows you to specify how a program can be reused.
FFMIDIQ	Char	12	FMID found by Inquisitor for module
FPTF	Char	8	Last PTF number for module
FPUTLEVEL	Char	8	PUTLEVEL that module was in
FRSU	Char	8	Last RSU level that module was in
FALIASNAME	Char	40	Alias name of module
FAPFAUTH	Char	1	APFAUTH flag Y or N

Table 50. TPARAM

Column name	Column type	Column length	Description
FKEY	Char	64	Parameter Key
FVALUE	Char	254	Parameter Value

Table 51. TPERIODS

Column name	Column type	Column length	Description
FPERIOD	Date		Calendar month for usage
FINVID	Integer		Inventory ID
FSUMMARISED	Smallint		Summary status

Table 52. 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

Table 52. TPOVINV (continued)			
Column name	**Column type**	**Column length**	**Description**
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 53. 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
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 54. TPRODUCT			
Column name	**Column type**	**Column length**	**Description**
FPRODUCTID	Integer		Product ID
FPRODUCTNAME	Char	80	Product Name (could be alias name)
FGLOBALNAME	Char	80	Product Name (always Global Name if Alias is used)
FOPTIONNAME	Char	80	Option Name
FVENDORID	Integer		Vendor ID
FPRODSTATUS	Smallint		Billable Status
FCATEGORY	Char	30	Product Category

Table 54. TPRODUCT (continued)

Column name	Column type	Column length	Description
FDESCRIPTION	Varchar	254	Product Description
FOBSERVEDELETED	Timestamp		Date and time that the library was deleted from Inquisitor data
FTYPE	Char	1	P=Product, M=Maybe, N=None, U=Unknown
FOWNER	Char	12	Owner of the product
FPRODUCTIDNAME	Char	12	Productidname is a designation given to a product that does not change even if the product is renamed
FOPSYS	Char	1	Operating system. Value Z or U

Table 55. TPRODUCT_REGISTRATION

Column name	Column type	Column length	Description
SID	Char	4	Product system ID. By default this is the SMFID
PID	Char	16	Product Identifier
PRODUCT_OWNER	Char	16	Product owner or vendor
PRODUCT_NAME	Char	50	Product name
PRODUCT_FEATURE	Char	16	Product feature or option name
PRODUCT_VERSION	Char	6	Product version
PRODUCT_FLAGS	Char	8	Register state entry flags
PERIOD	Date		Month for this entry

Table 56. TREGISTERED_PRODUCT_USAGE

Column name	Column type	Column length	Description
SID	Char	4	Product system ID. By default this is the SMFID
PRODUCT_OWNER	Char	16	Product owner or vendor
PRODUCT_NAME	Char	50	Product name
PRODUCT_VERSION	Char	8	Product version
PRODUCT_QUALIFIER	Char	8	Product qualifier
PERIOD	Date		Month for this entry
PID	Char	16	Product Identifier
PRODUCT_FLAGS	Char	8	Register usage entry flags
SMFRECORDS	Integer		SMF record count
TCBTIME	Integer		Product TCB time

Table 56. TREGISTERED_PRODUCT_USAGE (continued)			
Column name	**Column type**	**Column length**	**Description**
SRBTIME	Integer		Product SRB time

Table 57. TUSEMTD			
Column name	**Column type**	**Column length**	**Description**
FMTDID	Float		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
FJESID	Char	8	Last JES job ID updated for the month
FTCBCNT	Float		TCB count
FTCBTIME	Float		TCB time
FHWID	Smallint		NODE_KEY identifier

Table 58. TUSEPOV			
Column name	**Column type**	**Column length**	**Description**
FUSEPOVINVID	Float		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

Table 58. TUSEPOV (continued)			
Column name	**Column type**	**Column length**	**Description**
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
FSEQNO	Smallint		Internal use only
FACCCNT	Integer		Number of distinct account codes
FTCBCNT	Float		TCB count
FTCBTIME	Float		TCB time
FHWID	Smallint		NODE_KEY identifier

Table 59. TUSEPOVLIB			
Column name	**Column type**	**Column length**	**Description**
FUSEPOVLIBID	Float		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

Table 59. TUSEPOVLIB (continued)			
Column name	**Column type**	**Column length**	**Description**
FACCCNT	Integer		Number of distinct account codes
FTCBCNT	Float		TCB count
FTCBTIME	Float		TCB time
FHWID	Smallint		NODE_KEY identifier

Table 60. 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 61. 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 62. TVENDOR			
Column name	**Column type**	**Column length**	**Description**
FVENDORID	Integer		Vendor ID
FVENDORNAME	Char	50	Vendor Name (could be alias name)
FGLOBALNAME	Char	80	Reserved
FVENDORGUID	Char	32	Vendor globally unique ID
FOBSERVEDELETED	Timestamp		Date and time that the library was deleted from Inquisitor data

Table 62. TVENDOR (continued)			
Column name	**Column type**	**Column length**	**Description**
FALTVENDORNAME	Char	50	Alternate vendor name

Table 63. TVERSION			
Column name	**Column type**	**Column length**	**Description**
FPOVID	Integer		Version ID
FVERSIONNAME	Char	44	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
FOBSERVEDELETED	Timestamp		Date and time that the library was deleted from Inquisitor data
FFMID	Char	8	FMID of the z/OS
FSUITE_ID	Smallint		ID of the Suite
FPTF	Char	8	The PTF level of the product
FPUTLEVEL	Char	8	The PUTLEVEL of the product
FRSU	Char	8	The maintenance level of the product
FEOSDATE	Timestamp		End of Service Data
FGADATE	Timestamp		GA date of the product

Table 64. ACTIVCON			
Column name	**Column type**	**Column length**	**Description**
ACTIVCON_ID	Integer		Active Container ID
CONTAINER_ID	Char	6	ID of the Container
CONTAINER_NAME	Char	50	Name of the Container
SOLUTION_ID	Char	64	The solution ID supplied by IBM
MACHINE_ID	Char	6	Name of the Machine
SERIAL	Char	12	System z Hardware Serial
PERIOD	Date		Reporting Period
PEAK_4HOUR	Integer		Peak 4 Hour Consumption
TOTAL_MSU	Integer		Total MSU Consumption

Table 65. ACTIVPRD

Column name	Column type	Column length	Description
CONTAINER_ID	Char	6	ID of the Container
PID	Char	16	Product Identifier
SOLUTION_ID	Char	64	The solution ID supplied by IBM
PERIOD	Date		Reporting Period
PRODUCT	Char	80	Name of the Product
PRDTYPE	Char	8	MLC or IPLA
MACHINE_ID	Char	6	Name of the Machine
SERIAL	Char	12	System z Hardware Serial

Table 66. CONTCPU

Column name	Column type	Column length	Description
MACHINE_ID	Char	6	Name of the Machine
SERIAL	Char	12	System z Hardware Serial
MODEL	Char	10	Model of the Hardware - eg 2965-605
PERIOD	Date		Reporting Period
MSU	Integer		Total MSU for the machine

Table 67. CONTLPAR

Column name	Column type	Column length	Description
LPAR_ID	Char	20	Name of the LPAR
TOTAL_MINUTES	Integer		Total minutes LPAR could have been online
TOTAL_MINUTES_COLLECTED	Integer		Total minutes for LPAR collected
MISSING_MINUTES	Integer		Missing minutes between possible online and actual collected.
TOTAL_MSU_CONSUM	Integer		Total MSU Consumed
CONTAINER_ID	Char	6	ID of the Container
PERIOD	Date		Reporting Period
PEAK_HOUR_CONSUM	Integer		Peak Hour MSU consumption
MACHINE_ID	Char	6	Name of the Machine
SERIAL	Char	12	System z Hardware Serial

Table 68. CONTNER

Column name	Column type	Column length	Description
CONTAINER_ID	Char	6	ID of the Container
CONTAINER_NAME	Char	50	Name of the Container
SOLUTION_ID	Char	64	The solution ID supplied by IBM

Table 68. CONTNER (continued)			
Column name	**Column type**	**Column length**	**Description**
PERIOD	Date		Reporting Period
TOTAL_MSU	Integer		Total MSU consumed by the Container
MACHINE_ID	Char	6	Name of the Machine
SERIAL	Char	12	System z Hardware Serial

Table 69. EXCLUDE_SUITES			
Column name	**Column type**	**Column length**	**Description**
FLPARID	Integer		LPAR ID
FSYSPLEXID	Char	8	z/OS Sysplex name
PRODUCT_NAME	Char	80	Name of the Product
VERSION	Char	16	Version of the Product
PID	Char	16	PID of the Suite
SUITE_ID	Smallint		ID of the Suite
FOPSYS	Char	1	Operating System. Value Z or U

Table 70. PRODUCT_SUITES			
Column name	**Column type**	**Column length**	**Description**
FLPARID	Integer		LPAR ID
FSYSPLEXID	Char	8	z/OS Sysplex name
SUITE_ID	Smallint		ID of the Suite
SUITE_NAME	Char	80	Name of the Suite
SUITE_VERSION	Char	16	Version of the Suite
PID	Char	16	PID of the Suite
SW_KEY	Char	32	Global Unique ID (GUID) for this entry For SW_TYPE=VERSION this will be the same value as VERSION_GUID For SW_TYPE=FEATURE this will be the same value as FEATURE_GUID
VENDOR_NAME	Char	50	Vendor name of the Suite
FOBSERVEDELETED	Timestamp		Date and time that suite was deleted from system
FOPSYS	Char	1	Operating System. Value Z or U

Table 71. PRODUCT_VUE

Column name	Column type	Column length	Description
PRODUCT_NAME	Char	80	Product Name
FEATURE_NAME	Char	80	Feature Name
PID	Char	16	Product PID
VUPID	Char	16	VUE PID
VUEID	Char	8	VUE EID
VUSSPID	Char	8	VUE S&S PID
VUSSEID	Char	8	VUE S&S EID
VUICA	Char	1	VUE ICA
VUIPLA	Char	1	VUE IPLA

Table 72. TLOGAGGR

Column name	Column type	Column length	Description
FRUNDATE	Timestamp		Date and time Aggregator was run
FCOUNTUSAGEFULL	Char	1	Input value of parameter COUNTUSAGEFULL. Default value is N
FAPARLVEL	Char	10	APAR number
GKBVERSION	Char	6	Version of the GKB
FJOBSTEP	Char	15	Identify the last operational process (IQ Import or Usage Import) run before Aggregator.
FIMPORTDATE	Timestamp		Identify the last operational timestamp (IQ Import or Usage Import) run before Aggregator.

Table 73. TLOGLDEL

Column name	Column type	Column length	Description
FRUNDATE	Timestamp		Date and time LPAR deletion was run
FSID_PARM	Char	8	Input value of parameter SID
FAPARLVEL	Char	10	APAR number

Table 74. TLOGPDEL

Column name	Column type	Column length	Description
FRUNDATE	Timestamp		Date and time Physical deletion was run

Table 74. TLOGPDEL (continued)

Column name	Column type	Column length	Description
FFIRSTDATE	Char	6	Input value of parameter FIRSTDATE Input value for parameter RETENTION is also captured here For example: If RETENTION=0, then FFIRSTDATE=0 If RETENTION=12, then FFIRSTDATE=1
FLASTDATE	Char	6	Input value of parameter LASTDATE Input value for parameter RETENTION is also captured here For example: If RETENTION=0, then FLASTDATE=0 If RETENTION=12, then FLASTDATE=12
FAPARLVEL	Char	10	APAR number
FTLIBRARY_DEL	Integer		Number of records deleted from TLIBRARY table
FTLIBSYS_DEL	Integer		Number of records deleted from TLIBSYS table
FTMODULE_DEL	Integer		Number of records deleted from TMODULE table
FTPOVLIB_DEL	Integer		Number of records deleted from TPOVLIB table
FTCSECT_DEL	Integer		Number of records deleted from TCSECT table
FTVENDOR_DEL	Integer		Number of records deleted from TVENDOR table
FTVERSION_DEL	Integer		Number of records deleted from TVERSION table
FTPRODUCT_DEL	Integer		Number of records deleted from TPRODUCT table

Table 75. TLOGUDEL

Column name	Column type	Column length	Description
FRUNDATE	Timestamp		Date and time Usage deletion was run

Table 75. TLOGUDEL (continued)			
Column name	**Column type**	**Column length**	**Description**
FSID_PARM	Char	8	Input value of parameter SID (LPAR name or 'ALLSIDS')
FKEEPDETAIL	Smallint		Input value of parameter KEEPDETAIL. Default value is 2
FKEEPAGGR	Smallint		Input value of parameter KEEPAGGR. Default value is 12
FFIRSTDATE	Char	6	Input value of parameter FIRSTDATE
FLASTDATE	Char	6	Input value of parameter LASTDATE
FAPARLVEL	Char	10	APAR number
FTUSEMTD_DEL	Integer		Number of records deleted from TUSEMTD table

Table 76. TALTVENDOR			
Column name	**Column type**	**Column length**	**Description**
FVENDORNAME	Char	50	Vendor name
FALTVENDORNAME	Char	50	Alternate vendor name

Table 77. TLICENSE			
Column name	**Column type**	**Column length**	**Description**
LICENSE_KEY	INTEGER		License Key. Auto generated number
SCHEMA	CHAR	8	Schema
REPNAME	CHAR	80	Repository Name
SITE	CHAR	50	Optional Site Name
MACHINE	CHAR	12	Combination of hardware type and capacity model Sample: 002964-508Searila
SERIAL	CHAR	12	CPU Serial Number
LPAR	CHAR	20	LPAR name
LICENSE_TYPE	CHAR	10	Type of License, MLC or OTC
LICENSE_SCOPE	CHAR	10	Enterprise, Site, Machine, LPAR
LICENSE_DATE	DATE	4	YYYY-MM-DD format only
LICENSE_RNW_DATE	DATE	4	YYYY-MM-DD format only
ELA_START_DATE	DATE	4	YYYY-MM-DD format only
ELA_END_DATE	DATE	4	YYYY-MM-DD format only
VENDOR	CHAR	80	Vendor name
PRODUCT	CHAR	80	Product name

Table 77. TLICENSE (continued)

Column name	Column type	Column length	Description
FEATURE	CHAR	80	Feature name
VERSION	INTEGER		Version
PID	CHAR	16	PID
SSPID	CHAR	16	S&S PID
EIDPID	CHAR	16	Entitlement ID PID
CAPACITY_UNIT	CHAR	10	MSU, VU, MIPS, SUB. Currently only MSU is supported.
QUANTITY	INTEGER		Null or integer
INITIAL_PRICE	CHAR	20	Initial price
MAINT_PRICE	CHAR	20	Maintenance price
MAINT_PERCENTAGE	CHAR	3	Maintenance price percentage
VEND_CONT_NAME	CHAR	50	Vendor Contact Name
VEND_CONT_TITLE	CHAR	50	Vendor Contact Title
VEND_CONT_PHONE	CHAR	25	Vendor Contact Phone
VEND_CONT_EMAIL	CHAR	50	Vendor Contact Email
OWNER_CONT_NAME	CHAR	50	Owner Name
OWNER_CONT_TITLE	CHAR	50	Owner Title
OWNER_CONT_DEPT	CHAR	50	Owner Department
OWNER_CONT_PHONE	CHAR	25	Owner Phone
OWNER_CONT_EMAIL	CHAR	50	Owner Email
LICENSE_SERIAL	CHAR	255	License serial number

Table 78. LICENSE_VERIFY

Column name	Column type	Column length	Description
SCHEMA	CHAR	8	Schema name
REPNAME	CHAR	80	Repository name
IMPORT_ID	INTEGER		Unique value
IMPORT_DATE	TIMESTAMP		Date and time record was imported
SITE	CHAR	50	Site name
VENDOR_NAME	CHAR	80	Vendor Name
PRODUCT_NAME	CHAR	80	Product Name
FEATURE_NAME	CHAR	80	Feature Name
VERSION	INTEGER		Version
PID	CHAR	16	PID
VUE	CHAR	8	Value Unit PID

Table 78. LICENSE_VERIFY (continued)

Column name	Column type	Column length	Description
EID	CHAR	8	Entitlement ID
SYSPLEX	CHAR	8	Sysplex Name
SYSTEM	CHAR	4	LPAR Name
MSULPAR	INTEGER		MSU rating for LPAR
LICENSE_SCOPE	CHAR	20	Enterprise, Site, Machine, LPAR
MACHINE	CHAR	12	Hardware type, capacity, and serial number
SERIAL	CHAR	12	CPU Serial Number
LICENSE_DATE	DATE	4	Date of initial license
RENEWAL_DATE	DATE		Renewal date
LAST_USED	DATE		Date product was last used
SYSTEMS_USED	CHAR	2	How many LPARs used
MACHINES_USED	CHAR	2	How many machine used
LAST_UPDATE	DATE		Last date it was updated
ENTITLED	CHAR	1	Licensed Y or N
LICENSE_SERIAL	CHAR	255	License Serial Number
VERIFY_ERROR	VARCHAR	255	Verification errors

Table 79. TSTAGE

Column name	Column type	Column length	Description
LSTDATE	Char	10	Date of Usage
HDRSID	Char	4	System
DTLVOL	Char	8	Volume
DTLPATH	Char	128	Library
DTLMODNM	Char	40	Module Name
DTLJOB	Char	8	Job Name
DTLUSER	Char	8	User Name
DTLACCT	Char	20	Account ID
DTLDATE	Char	8	Date
DTLTIME	Char	6	Time
DTLJOBID	Char	8	Job ID
DTLCOUNT	INTEGER		Events
DTLTIME	INTEGER		Time
DTLJOBS	INTEGER		Job-Day Count
DTLTCBTM	INTEGER		Task CPU Time
DTLTONCP	INTEGER		General CP Time

Table 79. TSTAGE (continued)			
Column name	**Column type**	**Column length**	**Description**
DTLTOFFO	INTEGER		Offload Time
DTLACTIV	INTEGER		Active Time
DTLREAL	INTEGER		Real Name
DTLCALLR	INTEGER		Caller

Table 80. TNOTIFY			
Column name	**Column type**	**Column length**	**Description**
ID	INTEGER	10	ID Auto Generated
TEXT	VARCHAR	500	Notification text
SERVERITY	VARCHAR	20	Type of Notification
CREATED_AT	TIMESTAMP		Date notification created
EXPIRES_AT	TIMESTAMP		Expiry date of notification
IS_AUTOMATED	SMALLINIT		Flag indicating if the notification was automatically generated (e.g., by a program) or manually created.
UTIL_NAME	VARCHAR	50	Name of the program or utility that generated the notification (if automated).
USER_NAME	VARCHAR	50	Name of the user who created the notification (if manually produced).
SID	VARCHAR	4	LPAR name
IS_ACTIVE	SMALLINIT		Is notification valid

Post-installation jobs

After installation, make a copy of HSISCUST member from the hsi.SHSISAMP data set. Next, edit HSISCUST to create customized versions of any job in the JCLLIB library or any parameter in the PARMLIB library.

Jobs generated in JCLLIB for a Db2 environment

The custom JCLLIB members that you create with post-installation customization in a Db2 environment are used to submit jobs to implement the product.

Post installation jobs

The following table lists the post-installation jobs generated in the JCLLIB library when the DBTYPE=DB2.

Table 81. Post-installation jobs generated for a Db2 database	
Job	**Description**
HSISDB01	Job to define Db2 storage groups.
HSISDB02	Job to define the GKB database.

Table 81. Post-installation jobs generated for a Db2 database (continued)	
Job	**Description**
HSISDB03	Job to define the repository database.
HSISGKBL	Job to populate the GKB, GKB for z/OS UNIX, and Inquisitor filters.
HSISGRNT	Job to grant administrator access to the repository and GKB databases.
HSISGRTB	Job to grant SELECT access to the repository and GKB tables.
HSISANS1	Job to set up RACF security profiles in the Analyzer utility.
HSISANS2	Job to set up new SSL certificates in the Analyzer utility.
HSISANS3	Job to use existing SSL certificates in the Analyzer utility.

Operations jobs

The following table lists the operations jobs generated in the JCLLIB library when the DBTYPE is set to DB2.

Table 82. Operations jobs generated for a Db2 database	
Job	**Description**
HSISGKBL	Job to populate the GKB, GKB for z/OS UNIX, and Inquisitor filters. To be run when monthly updates are provided.
HSISINQZ	Job to run the Inquisitor.
HSISINQU	Job to run the Inquisitor for z/OS UNIX.
HSISIQIM	Job to run the Inquisitor Import for z/OS and z/OS UNIX.
HSIJMON	Started task - Usage Monitor.
HSISUMON	Job to run the Usage Monitor.
HSISUIMP	Job to run the Usage Import.
HSIJAUTO	Started task - Automation Server.
HSIASALC	Job to allocate the Automation Server control file.
HSIASSCT	Job to run the Automation Server Scout utility.

Reporting jobs

The following table lists the reporting jobs generated in the JCLLIB library when the DBTYPE is set to DB2.

Table 83. Reporting jobs generated for a Db2 database	
Job	**Description**
HSIJANLO	Started task - Analyzer online mode.

Table 83. Reporting jobs generated for a Db2 database (continued)	
Job	**Description**
HSISANLO	Job to run Analyzer reporting in online mode.
HSISANLB	Job to run Analyzer reporting in batch mode.

Utility jobs

The following table lists the Utility jobs generated in the JCLLIB library when the DBTYPE is set to DB2.

Table 84. Utility jobs generated for a Db2 database	
Job	**Description**
HSISZCAT	Job to concatenate and aggregate Usage Monitor data sets.
HSISPTAG	Job to run the Product Tagging utility.
HSISUDEL	Job to run the usage deletion.
HSISLDEL	Job to run the system deletion.
HSISPDEL	Job to run the physical deletion.
HSISMDEL	Job to run the MDEL deletion
HSISMDEL	Job to run the MDEL deletion
HSISLLST	Job to create an HLQ listing for the Usage Monitor.
HSISSCRT	Job to read SCRT CSV files and populate the repository.
HSISKBT	Job to run the XML utility. The XML output is for IBM Control Desk (ICD).
HSISIBM	Job to filter out non-IBM programs from the Inquisitor utility and usage data.
HSISPLTX	Job to create a PLT entry for CICS.
HSISENAX	Job to enable CICS GLUE program.
HSISUT01	Sample job to backup the repository database.
HSISUT02	Sample job to restore the repository database.
HSISUT03	Sample job to reorganize the repository database.
HSISUT04	Sample job to run the RUNSTATS utility on the repository database.
HSISIPVD	Diagnostic. Job to verify database changes since the product was released.
HSISTPRM	Diagnostic. Job to update the repository TPARAM table.

Table 84. Utility jobs generated for a Db2 database (continued)	
Job	**Description**
HSISCSI	Diagnostic. Job to gather SMP/E diagnostics data.
HSISSTAG	Job to get UMON/ZCAT data into a staging table.
HSISAPIP	Job to start up the REST API.

Jobs for porting data between repositories

The following table lists the jobs generated in the JCLLIB library for porting data between repositories when the DBTYPE=DB2.

Table 85. Jobs generated for porting data between repositories for a Db2 database	
Job	**Description**
HSISUN91	Job to unload the repository database.
HSISLO91	Job to load the repository database.

Migration jobs

The following table lists the migration jobs generated in the JCLLIB library when the DBTYPE is set to DB2.

Table 86. Migration jobs generated for a Db2 database	
Job	**Description**
HSISMS31	Job to verify database objects/counts.
HSISMS32	Job to update Repository TPARAM table, create new database objects from 8.2 to 9.1.
HSISMS35	Job to verify newly created database objects updated from HSISMS32.
HSISMS38	Job to update TPARAM table, create new database objects from 8.3/8.3.1 to 9.1.
HSISMS39	Job to verify newly created database objects updated from HSISMS38.

Globalization jobs

The following table lists the globalization jobs generated in the JCLLIB library when the DBTYPE is set to DB2.

Table 87. Globalization jobs generated for a Db2 database	
Job	**Description**
HSISMCMP	Job to compile Japanese messages for MMS.

Jobs generated in JCLLIB for a remote environment

The custom JCLLIB members that you create with post-installation customization in a remote environment are used to submit jobs to implement the product.

Operations jobs

The following table lists the operations jobs generated in the JCLLIB library when the DBTYPE=REMOTE.

Table 88. Operations jobs generated for a remote environment

Job	Description
HSISINQZ	Job to run the Inquisitor.
HSISINQU	Job to run the Inquisitor for z/OS UNIX.
HSIJMON	Started task - Usage Monitor.
HSISUMON	Job to run the Usage Monitor.
HSIJAUTO	Started task - Automation Server.
HSIASALC	Job to allocate the Automation Server control file.
HSIASSCT	Job to run the Automation Server Scout utility.

Utility jobs

The following table lists the Utility jobs generated in the JCLLIB library when the DBTYPE=REMOTE.

Table 89. Utility jobs generated for a remote environment

Job	Description
HSISZCAT	Job to concatenate and aggregate Usage Monitor data sets.
HSISPTAG	Job to run the Product Tagging utility.
HSISIBM	Optional. Job to filter out non-IBM programs from the Inquisitor and usage data.
HSISPLTX	Optional. Job to create a PLT entry for CICS.
HSISENAX	Optional. Job to enable CICS GLUE program.
HSISCSI	Diagnostic. Job to gather SMP/E diagnostics data.

Jobs generated in JCLLIB for an SQLite environment

The custom JCLLIB members that you create with post-installation customization in an SQLite environment are used to submit jobs to implement the product.

Post installation jobs

The following table lists the post-installation jobs generated in the JCLLIB library when the DBTYPE=SQLITE.

Table 90. Post-installation jobs generated for a SQLite database	
Job	**Description**
HSISDB01	Job to create SQLite zFS file system.
HSISDB02	Job to define the GKB database.
HSISDB03	Job to define the repository database.
HSISDB04	Job to define the Notifications database.
HSISGKBL	Job to populate the GKB, GKB for z/OS UNIX, and Inquisitor filters.
HSISGRNT	Job to grant access to z/OS OMVS groups.
HSISANS1	Job to set up RACF security profiles in Analyzer.
HSISANS2	Job to set up new SSL certificates in Analyzer.
HSISANS3	Job to use existing SSL certificates in Analyzer.

Operations jobs

The following table lists the operations jobs generated in the JCLLIB library when the DBTYPE is set to SQLITE.

Table 91. Operations jobs generated for a SQLite database	
Job	**Description**
HSISGKBL	Job to populate the Global Knowledge Base, Global Knowledge Base for z/OS UNIX, and Inquisitor filters. To be run when monthly updates are provided.
HSISINQZ	Job to run the Inquisitor.
HSISINQU	Job to run the Inquisitor for z/OS UNIX.
HSISIQIM	Job to run the Inquisitor Import for z/OS and z/OS UNIX.
HSIJMON	Started task - Usage Monitor.
HSISUMON	Job to run the Usage Monitor.
HSISUIMP	Job to run the Usage Import.
HSIJAUTO	Started task - Automation Server.
HSIASALC	Job to allocate the Automation Server control file.
HSIASSCT	Job to run the Automation Server Scout utility.

License verificaction jobs generated for an SQLite database

The following table lists the license verification jobs generated in the JCLLIB library when the DBTYPE is set to SQLite.

Table 92. License verificaction jobs generated for an SQLite database

Job	Description
HSISLICI	Job to run the License Import function
HSISLICIV	Job to run the License Verify function

Reporting jobs

The following table lists the reporting jobs generated in the JCLLIB library when the DBTYPE is set to SQLITE.

Table 93. Reporting jobs generated for a SQLite database

Job	Description
HSIJANLO	Started task - Analyzer online mode.
HSISANLO	Job to run Analyzer reporting in online mode.
HSISANLB	Job to run Analyzer reporting in batch mode.

Utility jobs

The following table lists the Utility jobs generated in the JCLLIB library when the DBTYPE is set to SQLITE.

Table 94. Utility jobs generated for a SQLite database

Job	Description
HSISZCAT	Job to concatenate and aggregate Usage Monitor data sets.
HSISPTAG	Job to run the Product Tagging utility.
HSISUDEL	Job to run the Usage Deletion.
HSISLDEL	Job to run the System Deletion.
HSISPDEL	Job to run the physical deletion.
HSISMDEL	Job to run the MDEL deletion.
HSISSCRT	Job to read SCRT CSV files and populate the Repository.
HSISIBM	Job to filter out non-IBM programs from the Inquisitor and usage data.
HSISPLTX	Job to create a PLT entry for CICS.
HSISENAX	Job to enable CICS GLUE program.
HSISUT01	Sample job to backup SQLite zFS file system.

Table 94. Utility jobs generated for a SQLite database (continued)	
Job	**Description**
HSISUT02	Sample job to restore SQLite zFS file system.
HSISIVPD	Diagnostic. Job to verify database changes since the product was released.
HSISTPRM	Diagnostic. Job to update the Repository TPARAM table.
HSISCSI	Diagnostic. Job to gather SMP/E diagnostics data.
HSISSTAG	Job to get UMON/ZCAT data into a staging table.
HSISSQLT	Diagnostic. Job to display SQLite database files.
HSISMNT	Diagnostic. Job to mount SQLite zFS file system and optionally delete the zFS file system.
HSISMNTU	Diagnostic. Job to unmount SQLite zFS file system after an IPL.
HSISCSI	Diagnostic. Job to gather SMP/E diagnostics data.
HSISSTAG	Job to get UMON/ZCAT data into a staging table.

Jobs for porting data between repositories

The following table lists the jobs generated in the JCLLIB library for porting data between repositories when the DBTYPE=SQLITE.

Table 95. Jobs generated for porting data between repositories for a SQLite database	
Job	**Description**
HSISUN91	Job to unload the repository database.
HSISLO91	Job to load the repository database.

Migration jobs

The following table lists the migration jobs generated in the JCLLIB library when the DBTYPE is set to SQLITE.

Table 96. Migration jobs generated for a SQLite database	
Job	**Description**
HSISMS31	Job to verify database objects/counts.
HSISMS32	Job to update Repository TPARAM table, create new database objects from 8.2 to 9.1.
HSISMS33	Job to migrate tables for altered columns.
HSISMS34	Job to drop temporary tables created from HSISMS33.
HSISMS35	Job to verify newly created database objects updated from HSISMS32.
HSISMS38	Job to update TPARAM table, create new database objects from 8.3/8.3.1 to 9.1.
HSISMS39	Job to verify newly created database objects updated from HSISMS38.

Globalization jobs

The following table lists the globalization jobs generated in the JCLLIB library when the DBTYPE is set to SQLITE.

Table 97. Globalization jobs generated for a SQLite database

Job	Description
HSISMCMP	Job to compile Japanese messages for MMS.

Database performance and tuning

You can perform various configurations to tune your database to provide the best performance for your IBM Z Software Asset Management environment.

Db2 performance and tuning

Various configuration options are available to assist you in optimizing performance for your environment.

Initial space allocation

This section is useful for the database administrator who must determine space requirements for IBM Z Software Asset Management. Listed in Table 98 on page 208 are guidelines for the initial spaces allocation based on the number of LPARs. The value for the SIZE parameter is specified in HSISCUST.

Table 98. Initial space allocation for the product

SIZE=	Initial space allocation	Number of LPARs
1	1600 cylinders	1-10
2	3750 cylinders	11-20
3	12600 cylinders	>20

In V8.2 , all table spaces and indexes are defined with 'COMPRESS Y'. For estimated space requirements of the 3 largest table spaces, refer to PARMLIB member HSISSQ17.

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 in order to minimize competition for mainframe resources in the Db2 production environments. To avoid competing for mainframe resources, run the jobs for the Inquisitor Import and Usage Import during off-peak periods. In addition, run the utility Usage Deletion during off-peak periods.

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, first investigate the buffer pools. Check with your site specialist on the types and size of buffer pools that are defined for this product.

For sites using Db2 data sharing, group Bufferpools must be defined with the same names.

Space allocation and utilization

In terms of space utilization, -1 has been set for all SECQTY to enforce Sliding Secondary Extents. This enables 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 requirements. Definitions for these Db2 objects are listed in the respective jobs in JCLLIB.

Repository tables with the biggest impact on performance due to size are TMODULE, TUSEMTD, and PRODUCT_USE_DETAIL. Data for the TMODULE table is populated during Inquisitor Import process. TUSEMTD, and PRODUCT_USE_DETAIL tables are populated during Usage Import. For example, you might have more than 300 million usage records in the TUSEMTD table, and more than 20 million modules identified in the TMODULE table. To minimize space utilization and improve SQL query

performance, you must prune your usage and obsolete records by running the Usage Deletion job HSISUDEL and Physical Deletion job HSISPDEL.

Declared Global Temporary tables
Declared Global Temporary tables are used during the Asset Aggregator process. The Work file table spaces must be large enough to handle this process. When the Aggregator job step is run, indexes on declared global temporary tables are created. By default, the bufferpool used by the index is dependent on the bufferpool defined for the Work file database. Parameter IXBUFFERPOOL in PARMLIB member HSISAGP1 can be used to substitute the default value.

Work file database
When you run some of the SQL queries, they can produce a large amount of output. In order to avoid any excessive output, increase the number and size of the table spaces in the work file database.

Reorganization and RUNSTATS
It is important to run reorganization of the Repository table spaces periodically, especially after Inquisitor Imports, Usage Imports, and Usage Deletion After reorganization of the Repository table spaces, it is also a good idea to run RUNSTATS for these table spaces.

SQLite performance and tuning

General zFS performance queries

zFS performance is dependent on many factors. To help you to optimize performance, zFS provides performance information to help determine bottlenecks. You can enter the following system commands to get information about the current operation of zFS:

- `F ZFS,QUERY,SETTINGS`
- `F ZFS,QUERY,ALL`

To query and reset performance counters, enter the following z/OS UNIX System Services command:

```
zfsadm
```

Resource Management Facility (RMF) support for zFS

RMF support is available for zFS. When you are considering zFS performance, investigate the zFS components that are involved in I/O processing to or from a zFS file system. In a shared file system environment it is better for performance if you can mount a file system as read-only rather than as read-write. If a file system is mounted as read-write, but is accessed primarily from a single system, such as SYS1, it can improve performance if that file system is z/OS UNIX owned on system SYS1.

You can also optimize zFS performance by tailoring the size of its caches to reduce I/O rates and path length. It is also important to monitor DASD performance to ensure that no disc volumes or channels are required to perform beyond the intended capacity.

Monitoring and tuning cache size to improve zFS performance

You can improve the performance of zFS by controlling the size of the caches that hold file system and log data. Monitor the following caches so that you can control them effectively to reduce I/O rates:

- The user file cache is used for all user files and performs I/O for all user files greater than 7 KB.
- The metadata cache performs I/O for all user files that are smaller than 7 KB.
- The log file cache stores file record transactions that describe changes to the file system.

Database resources used by IBM Z Software Asset Management

The following tables list the database resources. The PARMLIB members provide definitions for these database resources.

&GKBZSCHM_GKB7 database resources

Table 99. &GKBZSCHM_GKB7 database resources		
Qualifier	**Table name**	**Tablespace (UTS)**
&GKBZSCHM_GKB7	IQIDPGM	WIQ1GKB
&GKBZSCHM_GKB7	IQIDPOVID	WIQ2GKB
&GKBZSCHM_GKB7	PRODUCT	WPROGKB
&GKBZSCHM_GKB7	PRODUCT_SUITES	WPSUGKB
&GKBZSCHM_GKB7	TCOMPILER	WCPRGKB
&GKBZSCHM_GKB7	TMODELS	WMODGKB
&GKBZSCHM_GKB7	TPARAM	WPRMGKB
&GKBZSCHM_GKB7	TPRODLINK	WPDKGKB
&GKBZSCHM_GKB7	TPRODUCT	WPDTGKB
&GKBZSCHM_GKB7	TPRSMAP	WPSMGKB
&GKBZSCHM_GKB7	TSCORPAT	WSCPGKB
&GKBZSCHM_GKB7	TVENDOR	WVDRGKB
&GKBZSCHM_GKB7	TVERSION	WVERGKB

&GKBZSCHM_GKU7 database resources

Table 100. &GKBZSCHM_GKU7 database resources		
Qualifier	**Table name**	**Tablespace (UTS)**
&GKBZSCHM_GKU7	PRODUCT	WPROGKU
&GKBZSCHM_GKU7	PRODUCT_SUITES	WPSUGKU
&GKBZSCHM_GKU7	TPARAM	WPRMGKU
&GKBZSCHM_GKU7	TPRODUCT	WPDTGKU
&GKBZSCHM_GKU7	TSCORPAT	WSCPGKU
&GKBZSCHM_GKU7	TVENDOR	WVDRGKU
&GKBZSCHM_GKU7	TVERSION	WVERGKU

&GKBZSCHM_IQF7 database resources

Table 101. &GKBZSCHM_IQF7 database resources		
Qualifier	**Table name**	**Tablespace (UTS)**
&GKBZSCHM_IQF7	TCOMPILERS	WIQCPS
&GKBZSCHM_IQF7	TIQFILTERS	WIQFLS
&GKBZSCHM_IQF7	TPARAM	WIQPRM
&GKBZSCHM_IQF7	TXPCMODULES	WIQXMS
&GKBZSCHM_IQF7	TXPCSPEC	WIQXSC

Table 101. &GKBZSCHM_IQF7 database resources (continued)		
Qualifier	**Table name**	**Tablespace (UTS)**
&GKBZSCHM_IQF7	TXVENDORS	WIQXDR

&REPZSCHM_LKB7 database resources

Table 102. &REPZSCHM_LKB7 database resources		
Qualifier	**Table name**	**Tablespace (UTS)**
&REPZSCHM_LKB7	TPARAM	WPRMLKB
&REPZSCHM_LKB7	TPRODUCT	WPDTLKB
&REPZSCHM_LKB7	TSCORPAT	WSCPLKB
&REPZSCHM_LKB7	TVENDOR	WVDRLKB
&REPZSCHM_LKB7	TVERSION	WVERLKB

&REPZSCHM_LKU7 database resources

Table 103. &REPZSCHM_LKU7 database resources		
Qualifier	**Table name**	**Tablespace (UTS)**
&REPZSCHM_LKU7	TPARAM	WPRMLKU
&REPZSCHM_LKU7	TPRODUCT	WPDTLKU
&REPZSCHM_LKU7	TSCORPAT	WSCPLKU
&REPZSCHM_LKU7	TVENDOR	WVDRLKU
&REPZSCHM_LKU7	TVERSION	WVERLKU

&REPZSCHM database resources

Table 104. &REPZSCHM database resources		
Qualifier	**Table name**	**Tablespace (UTS)**
&REPZSCHM	ACTIVCON	VACTIVCN
&REPZSCHM	ACTIVPRD	VACTIVPR
&REPZSCHM	CONTCPU	VCONTCPU
&REPZSCHM	CONTLPAR	VCONTLPA
&REPZSCHM	CONTNER	VCONTNER
&REPZSCHM	EXCLUDE_SUITES	VEXCSUIT
&REPZSCHM	LICENSE_VERIFY	VLICENTL
&REPZSCHM	NODE	VNODE
&REPZSCHM	NODE_CAPACITY	VNODECAP
&REPZSCHM	PRODUCT	VPRODUCT
&REPZSCHM	PRODUCT_INSTALL	VPRODINS
&REPZSCHM	PRODUCT_NODE_CAPACITY	VPRNOCAP

Table 104. &REPZSCHM database resources (continued)		
Qualifier	**Table name**	**Tablespace (UTS)**
&REPZSCHM	PRODUCT_SUITES	VPRDSUIT
&REPZSCHM	PRODUCT_USE	VPRODUSE
&REPZSCHM	PRODUCT_USE_DETAIL	VPRODDET
&REPZSCHM	PRODUCT_VUE	VPRODVUE
&REPZSCHM	SYSTEM	VSYSTEM
&REPZSCHM	SYSTEM_NODE	VSYSNODE
&REPZSCHM	TACCOUNT	VACCOUNT
&REPZSCHM	TALTERNATE	VALTERNA
&REPZSCHM	TALTVENDOR	VALTVEND
&REPZSCHM	TANNOTATE	VANNOTAT
&REPZSCHM	TCHANNEL_PATH	VCHANNEL
&REPZSCHM	TCONTROL_UNIT	VCONTROL
&REPZSCHM	TCSECT	VCSECT
&REPZSCHM	TGADATES	VGADATES
&REPZSCHM	TISVEOS	VISVEOS
&REPZSCHM	TJOBDATA	VJOBDATA
&REPZSCHM	TLIBRARY	VLIBRARY
&REPZSCHM	TLIBSYS	VLIBSYS
&REPZSCHM	TLICENSE	VLICENSE
&REPZSCHM	TLOGAGGR	VLOGAGGR
&REPZSCHM	TLOGIQ	VLOGIQ
&REPZSCHM	TLOGLDEL	VLOGLDEL
&REPZSCHM	TLOGPDEL	VLOGPDEL
&REPZSCHM	TLOGUDEL	VLOGUDEL
&REPZSCHM	TLOGUI	VLOGUI
&REPZSCHM	TLPAR	VLPAR
&REPZSCHM	TMACHINE_RESOURCE	VMACHINE
&REPZSCHM	TMODULE	VMODULE
&REPZSCHM	TPARAM	VPARAM
&REPZSCHM	TPARAM	WPRMLKB
&REPZSCHM	TPARAM	WPRMLKU
&REPZSCHM	TPERIODS	VPERIODS
&REPZSCHM	TPOVINV	VPOVINV
&REPZSCHM	TPOVLIB	VPOVLIB

Table 104. &REPZSCHM database resources (continued)		
Qualifier	**Table name**	**Tablespace (UTS)**
&REPZSCHM	TPRODUCT	VTPRODUT
&REPZSCHM	TPRODUCT	WPDTLKB
&REPZSCHM	TPRODUCT	WPDTLKU
&REPZSCHM	TPRODUCT_REGISTRATION	VPRODREG
&REPZSCHM	TREGISTERED_PRODUCT_USAGE	VREGUSAG
&REPZSCHM	TSCORPAT	WSCPLKB
&REPZSCHM	TSCORPAT	WSCPLKU
&REPZSCHM	TSTAGE	VSTAGE
&REPZSCHM	TUSEMTD	VUSEMTD
&REPZSCHM	TUSEPOV	VUSEPOV
&REPZSCHM	TUSEPOVLIB	VUPOVLIB
&REPZSCHM	TUSEPRS	VUSEPRS
&REPZSCHM	TUSERDATA	VUSRDATA
&REPZSCHM	TVENDOR	VVENDOR
&REPZSCHM	TVENDOR	WVDRLKB
&REPZSCHM	TVENDOR	WVDRLKU
&REPZSCHM	TVERSION	VVERSION
&REPZSCHM	TVERSION	WVERLKB
&REPZSCHM	TVERSION	WVERLKU
&REPZSCHM	VWLOGIQ	VLOGIQ

Troubleshooting and support

To isolate and resolve problems with your IBM software, you can use the troubleshooting, messages, and support information. This information contains instructions for using the problem determination resources that are provided with your IBM products.

Troubleshooting a problem

Troubleshooting is a systematic approach to solving a problem. The goal of troubleshooting is to determine why something does not work as expected and how to resolve the problem.

The first step in the troubleshooting process is to describe the problem completely. Problem descriptions help you and the IBM technical-support representative know where to start to find the cause of the problem. This step includes asking yourself basic questions:

- What are the symptoms of the problem?
- Where does the problem occur?
- When does the problem occur?
- Under which conditions does the problem occur?

- Can the problem be reproduced?

What are the symptoms of the problem?

When starting to describe a problem, the most obvious question is "What is the problem?" This question might seem straightforward; however, you can break it down into several more-focused questions that create a more descriptive picture of the problem. These questions can include:

- Who, or what, is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, crash, performance degradation, or incorrect result?

The answers to these questions typically lead to a good description of the problem, which can then lead you a problem resolution.

Two main ways to approach any problem you encounter are understanding messages and using log files.

Messages

Messages are issued when unexpected events occur. Messages can have any of these severities:

Informational
The message confirms an event that was requested or describes another normal occurrence. Informational messages generally do not require any action. The identifier of an informational message ends with the letter I.

Warning
The message describes an event that might indicate a problem. Read the message text and determine whether the event is normal or a problem. The identifier of a warning message ends with the letter W.

Error
The message describes an event that requires a response. Read the message description and the suggested response. The identifier of an error message ends with the letter E.

Severe Error
The message describes an event that requires a response. Read the message description and the suggested response. The identifier of an error message ends with the letter S.

Unrecoverable Error
The message indicates that an unrecoverable error was encountered and no requests were processed. Read the message description. The identifier of an error message ends with the letter U.

You can find a message description easily by entering its identifier into the Search box in the information center.

Problems and solutions

Solution information helps you to understand the causes of an issue with your product and learn what to do to diagnose or resolve the problem.

Resolving Db2 SQLCODE -805 error for DSNACLI plan

If you receive a SQLCODE -805 error, operational jobs fail.

Symptoms

IBM Z Software Asset Management operational jobs are failing.

Causes

The DSNACLI plan is not bound with the latest Db2 maintenance package, or the plan references a missing package.

Diagnosing the problem

Check the log files for error messages in the IBM Z Software Asset Management operational jobs. The following log entry provides an example of the failure of the DSNCLIQR package:

```
Native Error Code-805
{Db2 FOR OS/390}{ODBC DRIVER}{DSN10015}
 DSNT408I SQLCODE = -805,
ERROR:  DBRM OR PACKAGE NAME DBA2..DSNCLIQR.18F920E31-
          3F8F1D9
NOT FOUND IN PLAN DSNACLI. REASON 03
```

Resolving the problem

Rerun db2.SDSNSAMP(DSNTIJCL) for all packages that the DSNACLI plan requires to be bound at the same time. The list of packages is slightly different for each release of Db2 for z/OS.

Ensure that the DSNAOCLI package has the following parameters:

```
 BIND PACKAGE (DSNAOCLI) MEMBER(DSNCLIMS) -
    CURRENTDATA(YES) ENCODING(EBCDIC)
    SQLERROR(CONTINUE)
```

Insufficient space in the Db2 work file database

When processing large amounts of data, you can encounter insufficient space in the Db2 work file database.

Symptoms

IBM Z Software Asset Management operational jobs are failing.

Causes

When you run SQL queries that process large amounts of data, including sorts and joins, there is insufficient space in the work file database that is used for temporary storage.

Diagnosing the problem

Check for examples of following messages in the Db2 MSTR address space:

```
DSNT501I  -DE91 DSNIXWKF RESOURCE UNAVAILABLE  553
           CORRELATION-ID=XXXXXX
           CONNECTION-ID=DB2CALL
           LUW-ID=AUIBMQXP.OMU1DEC1.C4729058740C=0
           REASON 00C900A5
           TYPE 00000230
           NAME 4K
DSNT501I  -DE91 DSNIWKFL RESOURCE UNAVAILABLE  554
           CORRELATION-ID=XXXXXX
           CONNECTION-ID=DB2CALL
           LUW-ID=AUIBMQXP.OMU1DEC1.C4729058740C=0
           REASON 00C90084
           TYPE 00000230
           NAME 4K
```

View and check the sizes and extents of the physical data sets allocated to the table spaces in the work file database.

Resolving the problem

Increase the sizes of the table spaces associated with the work file database. The sample db2.SDSNSAMP(DSNTIJTM) job provides examples of how to increase the sizes of the table spaces.

Preventing Db2 timeouts/deadlocks during Inquisitor or Usage imports

Batch import jobs from the Inquisitor or the Usage Monitor require exclusive access to tables in the Repository database to prevent timeouts or deadlocks.

Symptoms

IBM Z Software Asset Management operational jobs for Inquisitor or Usage import are failing.

Causes

When running Inquisitor Import or Usage import batch jobs, multiple users using the Analyzer to access the Repository database can lead to timeouts or deadlocks. These batch jobs require exclusive accesses to the data in the tables. SQLCODE -904 error occurs with reason code 00C90083 or reason code 00C9008E

Diagnosing the problem

Check for examples of following messages in the Db2 MSTR address space:

```
DSNT501I  - DSNIDBET RESOURCE UNAVAILABLE  656
            CORRELATION-ID=AAAAAA
            CONNECTION-ID=DB2CALL
            LUW-ID=NETA.GGGGGG.UUUUD99=81188
            REASON 00C90083
            TYPE 00000200
            NAME XXXXXX.YYYYY

DSNT501I  - DSNIDBET RESOURCE UNAVAILABLE  656
            CORRELATION-ID=BBBBB
            CONNECTION-ID=DB2CALL
            LUW-ID=NETA.GGGGGG.IIIII222=81188
            REASON 00C9008E
            TYPE 00000200
            NAME XXXXXX.YYYYY
```

Resolving the problem

You can perform one or more of the following changes to resolve the issue:

- Run these jobs during off-peak periods.
- Reduce the number of users that need to use the Analyzer during peak periods.
- Reduce the value of the COMMIT=1000 attribute to COMMIT=500 in PARMLIB members HSISIQP1 and HSISUIP1.
- Define the DSNACLI plan and the dependent packages to use uncommitted reads.
  - Modify all packages and plan in db2.SDSNSAMP(DSNTIJCL) to have ISOLATION(UR) and then run the job.
  - When you change the settings for DSNACLI, ensure that this plan is not used by other applications or create the plan with a different name.
  - By setting ISOLATION(UR), when you run Analyzer reports, it is likely that the correct or latest information is not displayed due to concurrency issues in Db2.

Improving Db2 performance

For Db2 sites, IBM Z Software Asset Management operational jobs are usually run during off-peak hours, so it is important that all IQ Import and Usage Import jobs must complete during the batch window.

Symptom

- Operational jobs such as IQ Import and Usage Import seem to be taking longer to run.
- Reports retrieved via the Analyzer also seem to take longer to finish.

Causes

This issue can occur for the following reasons:

- Retaining too much usage data for long periods and not performing regular housekeeping tasks.
- Importing all usage data into a single repository instead of spreading the usage data across multiple repositories.

Diagnosing the problem

- Identify operational jobs that are now taking longer to run.
- Identify Analyzer reports that are now taking longer to finish.
- In the Db2 MSTR address space/system console, check for any DSNJ031I or DSNR035I messages, indicating that jobs may not be committing at regular intervals as expected.
- In the Db2 MSTR address space/system console, check for problems related to the Db2 work file database (this has been covered in a previous problem/solution).
- Run SQL command to determine size and periods of the largest table – TUSEMTD.
- Run SQL command to list libraries that are marked deleted (obsolete).
- IBM Support may request you to run special diagnostics/traces.
- Run SQL command to determine size and periods of the largest table – TUSEMTD.

```
SELECT FPERIOD, COUNT(*) FROM &RESPZSCHM.TUSEMTD
GROUP BY FPERIOD ;
```

  Next, review comments in job HSISUDEL and decide on range of periods to select for deletion of usage records.

- Run SQL command to list libraries that are marked deleted (obsolete).

```
SELECT DISTINCT CAST(YEAR(FOBSERVEDELETED)AS CHAR(4))||
'-'||CASE WHEN MONTH(FOBSERVEDELETED) < 10
THEN '0'
ELSE '' END || CAST(MONTH(FOBSERVEDELETED)
AS CHAR(2))
FROM &REPZSCHM.TLIBRARY
WHERE FOBSERVEDELETED IS NOT NULL;
```

  The above SQL command is found in job, HSISPDEL. Review comments in the job and decide on range of periods to select for deletion of obsolete records from the TLIBRARY, TLIBSYS, TMODULE, TCSECT, TPOVLIB, TPRODUCT, TVENDOR, and TVERSION tables.

Resolving the problem

For Analyzer reports, especially from the Discovery Tab, do not use an '*' as the first character as a wild card. For example, in report 'Search by Modules, restrict your search to a more definitive range.

As part of improving the run times, you may be advised by IBM Support to change the value for the COMMIT parameter required in the IQ Import and Usage Import jobs.

- Run job HSISLDEL to delete obsolete LPARs (LPARs that have been decommissioned at your site).

- For sites that have an extremely large TUSEMTD table (50m to 100m rows per period), consider defining the table space as a partition-by-range (PBR) UTS. Use column FPERIOD in TUSEMTD table as the key for the range. As this is not a trivial task, the DBA needs to prepare an implementation plan for this exercise to convert from a PBG to a PBR.
- On a regular basis, run REORG for all repositories. RUNSTATS is automatically run as part of IQ Import and Usage Import jobs.

Updating your Global Knowledge Base (GKB) database

IBM support provides monthly update to the Global Knowledge Base (GKB), so that you can keep your product inventory definitions up-to-date. You can also submit items to IBM support for inclusion in GKB updates.

Symptoms

Products displayed in the Analyzer reports are not correct.

Causes

The GKB database does not contain the latest updates.

Diagnosing the problem

To verify the latest version of the GKB database, check the log in HSISIQIM job for GKB Version = *yymmdd*. The *yymmdd* variable represents the version of GKB monthly update that is applied at your site.

Resolving the problem

Refer to the section "Updating the Global Knowledge Base" on page 49 on how to download and update the latest copy of the GKB database.

IBM Z Software Asset Management messages

You can identify the type of message by the message prefix.

HSIA - Automation Server messages

Return codes

Table 105. Return codes and their meaning

Return code	Description
0	No errors encountered. All requests processed successfully.
16	Unrecoverable error. No requests processed. SYSIN or HSAPZIP or INQSOUT File cannot be used, or unsupported operating system.

Message suffix codes

Table 106. Message suffix codes and associated condition codes

Suffix	Meaning	Raises minimum condition code to:
I	Information message	0
W	Warning message	4

Table 106. Message suffix codes and associated condition codes (continued)		
Suffix	**Meaning**	**Raises minimum condition code to:**
E	Error message	8
S	Severe error message	12
U	Unrecoverable error message	16, 20

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

section:
Parsing did not detect the expected close parenthesis.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA002E EXPECTED VALUE FOR *parm* NOT FOUND BEFORE THE CLOSE PARENTHESIS

section
No subparameter value was specified within the parentheses.

In the message text:

parm
name of parameter being processed.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA003E THE *parm* PARAMETER IS NOT RECOGNIZED

section
A parameter was detected which is not valid for the type of statement being processed.

In the message text:

parm
name of parameter being processed.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO

HSIA004E THE VALUE SPECIFIED FOR *parm* IS INVALID

section
The named parameter had a value which did not conform to syntax requirements.

In the message text:

parm
name of parameter being processed.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO

HSIA005E **NO "FTP" OR "JOB" STATEMENT BEFORE "DSN" STATEMENT**

section:
There is no preceding action to relate the dsname mask to.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA006E ***parm*** **IS AN UNRECOGNIZED STATEMENT TYPE**

section

A statement type other than FTP, JOB, or DSN was specified.

In the message text:

parm
name of parameter.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA007E **TERMINATING - AUTOMATION SERVER IS ALREADY ACTIVE**

section:
The Automation Server is already running. Only one concurrent copy can run in an operating system image.

section:
Program terminates with condition code 16. The established Automation Server continues.

section:
None.

section:
None.

section:
HSIAUTO

HSIA008E **TERMINATING - COULD NOT INITIALISE WITH BAD PARAMETERS**

section:
The HSIAPARM contents contained an error so the Automation Server could not initialize.

section:
The program terminates with condition code 16.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA009E **REFRESH IGNORED - COULD NOT PROCESS BAD PARAMETERS**

section:
The HSIAPARM contents contained an error so the Automation Server could not update its operational parameters.

section:
Terminates the processing of HSIAPARM contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA010E **NO FUNCTIONS WERE REQUESTED**

section:
No actions were specified. The Automation Server has no work to do.

section:
Terminates the processing of HSIAPARM contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA011E **NO DATA SET NAME MASKS WERE SPECIFIED**

section:
No work was requested. The Automation Server has no work to do.

section:
Terminates the processing of HSIAPARM contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA014E MEMBER *mbr* WAS NOT FOUND IN THE PARMLIB FILE

section

Member HSIAPARM was found to be missing from the PARMLIB file.

In the message text:

mbr
name of missing member.

section:
Terminates the processing of the member. If the member is HSIAPARM the Automation Server terminates. For template members the Automation Server continues processing.

section:
Create the required member in the PARMLIB library.

section:
None.

section:
HSIAUTO HSIADSN

HSIA015E INPUT LOGICAL RECORD LENGTH WAS NOT 80

section:
A record was read from the PARMLIB library which was not 80 bytes long.

section:
The program terminates and takes no actions.

section:
Ensure the PARMLIB library only contains fixed length 80-byte records.

section:
None.

section:
HSIAUTO HSIADSN

HSIA016E EXPECTED OPEN PARENTHESIS WAS NOT FOUND IN INPUT RECORD

section:
Parsing did not detect the expected open parenthesis.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO

HSIA017E VALUE SPECIFIED FOR *parm* IS TOO LONG

section

A parameter value was specified which has a length greater than the maximum allowed for the named parameter.

In the message text:

parm
name of parameter.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO HSIADSN

HSIA018E THE "NOTA" VALUE IS LESS THAN THE "NOTB" VALUE

section:
The action can never be performed because all days of the month have been excluded by the combination of the NOTA (not after) and NOTB (not before) specifications.

section:
Terminates the processing of the HSIAPARM member contents.

section:
Correct the HSIAPARM member contents.

section:
None.

section:
HSIAUTO

HSIA019I AUTOMATION SERVER INITIALIZATION COMPLETE

section:
The Automation Server is commencing normal operations.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIAUTO

HSIA020I **AUTOMATION SERVER HAS NOW TERMINATED**

section:
The Automation Server is ceasing normal operations.

section:
Processing concludes.

section:
None.

section:
None.

section:
HSIAUTO

HSIA999U **HSIMSG/HSIAMSG FAILURE - MSGID=*msgid* RC=*rc* RS=*rs***

section
HSIMSG was called to produce a message text, but the call failed.

In the message text:

msgid
identifier of the failing message.

rc
HSIMSG return code.

rs
HSIMSG reason code.

section:
Terminates with a condition code of 20.

section
Inform the system programmer.

section
Ensure JOBLIB/STEPLIB contains the library where the HSIAMSG message module resides. If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM

support.

section:
HSIAUTO

HSIC - Operation messages

These messages are issued by the HSIC - Computation phase.

Return codes

Table 107. Return codes and their meaning

Return code	Description
0	No errors encountered. All requests processed successfully.
16	Unrecoverable error.

Message suffix codes

Table 108. Message suffix codes and associated condition 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

Table 108. Message suffix codes and associated condition codes (continued)		
Suffix	**Meaning**	**Raises minimum condition code to:**
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**

section:
A message is missing from the internal message repository. When the default language is not English, the translation of the given message might not exist. If the default language is English, the message indicates an error in the given application.

section:
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.

section:
Contact IBM support.

HSIC003U **The internal message repository is corrupted**

section:
When attempting to issue a message, the internal message repository layout did not follow the expected format.

section:
The application terminates.

section:
Contact IBM support.

HSIC020E ***application-name* encountered errors. Error code = *errorcode***

section:
The Application has encountered errors during processing. This is a general message on completion indicating that an error has occurred.

section:
Completes with given error code.

section:
Refer to additional message, or to the section 6016, and to the log for more details on the specific error. Contact IBM support.

HSIC021S ***application-name* encountered fatal errors. Error code = *error-code***

section:
The application has encountered fatal errors during processing.

section:
Terminates with given error code.

section:
Refer to additional message, or to the section 6016, and to the log for more details on the specific error. Contact IBM support.

HSIC023E **Inquisitor Import error occurred in opening: *filename***

section:
The Inquisitor Import could not open the given file.

section:
Terminates without processing any records.

section:
Check that the file exists, and if it does, check for any additional log message identifying the error. Contact IBM support.

HSIC024E **Inquisitor Import input file is in error. It looks like a usage data file**

section:
The Inquisitor Import has encountered an invalid input file.

section:
Terminates without processing any records.

section:
Check that the input file is a valid file. Contact IBM support.

HSIC025E **Inquisitor Import input file is in error. It looks like a hardware data file**

section:
The Inquisitor Import has encountered an invalid input file.

section:
Terminates without processing any records.

section:
Check that the input file is a valid file. Contact IBM support.

HSIC026E **Inquisitor Import detected that table *tablename* is missing or invalid**

section:
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.

section:
Terminates without processing any records.

section:
Check for a version mismatch between the database and the version of the product. Contact IBM support.

HSIC027S **Inquisitor Import table *tablename* is missing a column**

section:
The given table is missing an expected column. This suggests a mismatch between the database and this version of the product.

section:
The application terminates without processing any records.

section:
Check for a version mismatch between the database and the version of the product. Contact IBM support.

HSIC028S **Inquisitor Import table *tablename* appears to be an old version**

section:
The given table in the database does not have the expected format.

section:
The application terminates without processing any records.

section:
Check for a version mismatch between the database and the version of the product. Contact IBM support.

HSIC029S **Inquisitor Import error when writing to table *tablename***

section:
An SQL error occurred when attempting to write to the given table.

section:
The application terminates.

section:
Check the log for additional details about the given error. Contact IBM support.

HSIC030S **The Inquisitor Import did not find a valid system header record in the input file**

section:
The input file does not follow the expected format.

section:
The application terminates.

section:
Check that the correct input file is supplied and that there is no version mismatch. Contact IBM support.

HSIC034S **Error reading Repository TPARAM table**

section:
An error occurred while reading the TPARAM Repository table.

section:
The application terminates.

section:
Check the log for any additional messages indicating the cause of the error. Contact IBM support.

HSIC035E **The Repository is in use by the *application-name***

section:
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 supplied HSISTPRM job to reset FVALUE to 0 where FKEY = PROCRUN in the TPARAM table.

section:
The application terminates.

section:
Check that the application is not already in use before running this application.

HSIC036E **Syntax error scanning TPARAMS file on line *linenumber***

section:
The TPARAM file does not conform to the required syntax on the given line.

section:
The specified option or value is ignored, and its default value is used where applicable.

section:

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***

section:
A schema ID that is too long has been specified.

section:
The application terminates.

section:
Check that the schema ID does not exceed 8 characters in length.

HSIC038E **Unbalanced quote for value: *value* in param:*param***

section:
A starting quote that has no matching end quote was found for the given parameter.

section:
The application terminates.

section:
Check that the given parameter has matching quotes.

HSIC039E **Illegal character in value:*value* of param:*param***

section:
An invalid character was found in the given value.

section:
The application terminates.

section:
Check that the given parameter value is valid for its type.

HSIC040E **Reserved word: *reservedword* in param: *param***

section:
A reserved word or system value schema ID was chosen as a parameter value.

section:
The application terminates.

section:
Specify a different parameter value.

HSIC041W **value:*value* in param:*param* is not a recommended schema ID**

section:
The value is not recommended because of possible conflicts with existing values.

section:
The application continues.

section:
Choose a different value to avoid conflicts.

HSIC042E **TPARAM file: param:*param* has an invalid proposed value: *value***

section:
The parameter cannot be set to the given value because the value is not valid.

section:
The value is ignored, and the application continues.

section:
Choose a valid value as per the documentation of the given application.

HSIC043E **The application has failed to open the TPARAM file. Error: *errordescription***

section:
The application could not open the TPARAM file. The error description contains more details regarding the reason for the error.

section:
The application terminates.

section:
Check that the TPARAM file exists and is valid.

HSIC045E **String *string* cannot exceed *numberchars* in length**

section:
A parameter length limit has been exceeded.

section:
The application terminates.

section:
Ensure that the specified parameter length is not exceeded.

HSIC050E **The *program-name* program has detected an invalid date parameter**

section:
A date parameter was found to be invalid.

section:
The application terminates.

section:
Ensure that the date format is valid, and start dates do not overlap end dates.

HSIC051S **Error adding record**

section:
An SQL error occurred when adding a record to a table.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC052S Error updating record

section:
An SQL error occurred when updating a record in a table.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC053S Error deleting record

section:
An SQL error occurred when deleting a record from a table.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC055S Table initialization failure during Repository Merge

section:
At least one table initialization failed when merging repositories.

section:
The application terminates.

section:
Check the log for any additional details about this error. Contact IBM support.

HSIC056S Some table destination fields are smaller than source

section:
Some fields in the target repository are not large enough to fit the contents of fields in the source repository.

section:
The application terminates, and the repositories are not merged.

section:
Check that the destination repository is not an older version than the source repository. You can recreate the destination repository using the latest version of the product. If the problem persists, contact IBM support.

HSIC057E A value for parameter: *parameter-name* must be specified

section:
A mandatory parameter for this application has not been specified.

section:
The application terminates during the syntax checking of input parameters.

section:
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.

HSIC058E Could not open *filename*

section:
File could not be opened.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC059E Could not read *filename*

section:
File could not be read.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC060E IQDATA DD does not contain unzipped IQ data

section:
The input IQDATA dataset does not contain unzipped IQ data.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC061E Internal error hcreate*(number)* phase1a failed

section:
An internal error has occurred.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC062E No SMF 30-2 or 30-4 data matched IQ data

section:
No match was found for the SMF data and IQ data.

section:

The application terminates.

section:
Check that the correct data sets have been used. Contact IBM support.

HSIC063E Internal error hsearch*(key)* table add failed

section:
An error occurred when inserting data into a table.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC064E Could not write *type* to FMOUT

section:
Could not write to file FMOUT.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC065E SYSUT1 data Is not IQ text or UM text

section:
The SYSUT1 dataset does not contain the expected data.

section:
The application terminates.

section:
Check that the SYSUT1 dataset is correct. Contact IBM support.

HSIC066E Internal error hsearch*(key)* table failed

section:
An error occurred when retrieving data from a table.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC067E Unable to acquire storage

section:
An error has occurred when attempting to acquire storage.

section:
The application terminates.

section:
Try increasing the region size specified in the region parameter on the JOB or EXEC statement in the JCL for the job. Contact IBM support.

HSIC068E IBMMOD Internal error

section:
An internal error has occurred.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC069E IBMMOD_INIT internal error

section:
An error occurred when retrieving data from a table.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC070I A full rematch will be performed

section

A full import and rematch will be performed, which will not try to exclude modules of unchanged libraries. The default behavior is to exclude such libraries from matching, which would normally lead to faster processing.

The program performs a full rematch, if any of the following is true :

- If requested by the FULLREMATCH option.
- When the specified inventory is not found, for example on the first run when the inventory has not yet been created, and no previous match was done.
- If it is safer to perform a full rematch, as when a GKB change is detected or the REPLACEFULL option is in effect.

More specific details on why a full rematch is being performed, can be found in the log.

section:
A full rematch of the data is performed. All libraries are processed.

section

Ensure that a FULLREMATCH and the REPLACEFULL options are not in effect for better performance, unless a full rematch is desired.

If this is the first run of the Inquisitor Import, or there has been recent a change to the GKB, then no action is necessary; the program will try on subsequent

runs (subsequent to loading the current data into the repository) to exclude unchanged libraries.

HSIC071I ***&number_modules*** **modules in** ***&number_libraries*** **unchanged libraries were ignored.**

section:
This is a report of the number of the modules and libraries that are ignored when the FULLREMATCH option is not in effect. Details of these ignored libraries are in the log.

section:
None.

section:
None.

HSIC073E **Usage Import for system** ***SID*** **on** ***time*** **is a duplicate and will be ignored.**

section:
This input file has already been processed.

section:
The file is not processed. Processing is terminated.

section:
Provide an input file that is the output of a more recent Usage Monitor output.

HSIC074E **IQ input file dated** ***time1*** **is earlier than the latest** ***os_type*** **scan for SID** ***sid*** **dated** ***time2*****.**

section:
The input file is earlier than an input file that was already processed for this system.

section:
The file is not processed. Processing is terminated.

section:
Provide an input file that is the output of a more recent Inquisitor scan.

HSIC075I ***num_libs*** **libraries containing** ***num_modules*** **modules are shared.**

section:
The program encountered shared libraries, which are libraries that are identical in name and volume to libraries that the program previously processed on different SIDs.

section:
The program records the names and locations of such libraries but does not process their contents of modules. The program displays the names of such libraries in the log, as well as their current and base SIDs.

section:
This operation is normal as long as the repository is set up correctly to receive SIDs containing libraries that are unique in library and volume name except when they are copies or shared, and no other SIDs that do not conform to these rules were mistakenly placed into the repository. Refer to the Inquisitor Import description for more information. Refer to the log for a list of shared libraries and their base SIDs. You can run the System Deletion Job to remove any SIDs that have been placed recently but incorrectly into this repository.

HSIC076I **IQ input file dated** ***time*** **is a duplicate of the last** ***OS_TYPE*** **scan for SID** ***SID_NAME*****.**

section:
This input file has already been processed.

section:
The program continues and the file is processed.

section:
None.

HSIC077I **Analyzer initialization complete.**

section:
The Analyzer has started.

section:
Processing continues.

section:
None.

HSIC078I **Analyzer has now terminated.**

section:
The Analyzer has stopped.

section:
Processing continues.

section:
None.

HSIC079I **Analyzer is unable to display URL because gethostname returned a null string for the host name.**

section:
The analyzer attempted to get the host name of the system it is running on using the z/OS C/C++ gethostname library function and a null string was returned. This may happen if the TCP/IP TCPIP.DATA configuration file does not have a HOSTNAME statement.

section:
The analyzer is unable to display the url that clients can use to access it in its log. The analyzer continues processing and listening for connections from clients.

section:

Add a HOSTNAME statement to the TCP/IP TCPIP.DATA configuration file. In order for this change to have an effect, the TCP/IP address space must be stopped and restarted. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information on the HOSTNAME statement, on the TCPIP.DATA configuration file, and how to modify them.

HSIC080I **Analyzer is unable to display URL because getaddrinfo for host *hostName* failed with errno *errnoNumericValue*, *errnoDescriptionString*.**

section:
The analyzer attempted to get the fully qualified domain name and IP address for the host it is running on, which is named *hostName*. The z/OS C/C++ getaddrinfo library function was used, and it returned the error number shown.

section:
The analyzer is unable to display the URL that clients can use to access it in its log. The analyzer continues processing and listening for connections from clients.

section:
Refer to the *z/OS Communications Manager: IP and SNA Codes* manual for a description of Resolver return codes. It is possible that the host name could not be resolved due to a Resolver configuration problem or a Domain Name System (DNS) configuration problem. Refer to *z/OS Communications Server: IP Configuration Guide* and the *z/OS Communications Server: IP Configuration Reference* for information about how to configure Resolver and the BIND 9-based Domain Name System.

HSIC081E **Incompatible GKB level for gkb schema:*schema*. Expected level:*expected_level_number*, received::*received_level_number*.**

section

A mismatch between the GKB level and the level expected by the program was found. This indicates that the GKB and the code are at different maintenance levels. If the *received_level_number* is greater than the *expected_level_number*, it indicates that a later level of the GKB is used than can be handled by the current maintenance level of the code.

If the *received_level_number* is less than the *expected_level_number*, it indicates that an old level of the GKB, that can no longer be handled by the current maintenance level of the code, is used.

section:
The application terminates.

section:
Apply GKB and code maintenance to the indicated GKB schema as required to ensure code and GKB are compatible. Contact IBM support for further assistance.

HSIC082E **Incompatible repository level for repository schema: *schema*. Expected level:*expected_level_number*, received::*received_level_number*.**

section

A mismatch between the Database repository level and the level expected by the program was found. This indicates that the repository and the code are at different maintenance levels. If the *received_level_number* is greater than the *expected_level_number*, it indicates that a later level of the repository is used than can be handled by the current maintenance level of the code.

If the *received_level_number* is less than the *expected_level_number*, it indicates that an old level of the repository, that can no longer be handled by the current maintenance level of the code, is used.

section:
The application terminates.

section:
Apply Database repository and code maintenance to the indicated repository schema as required to ensure code and repository are compatible. Contact IBM support for further assistance.

HSIC083I **GKB schema: *schema* of version date:*gkb_date* is out of date. It is more than *number of months* old.**

section:
The used GKB of the given schema name is likely to be superseded and can be replaced by a more recent GKB containing more product updates. The GKB that is used is older than the default value of the *number_of_months* from the current date which means the latest identification changes may be missed.

section:
The application continues.

section:
Apply GKB maintenance as soon as possible.

HSIC084E **GKB schema: *schema* is missing configuration parameters.**

section:
The used GKB of the given schema name is missing configuration parameters that the program needs. This

is most likely to occur if the GKB has not been loaded successfully.

section:
The application terminates.

section:
Ensure GKB is loaded successfully. Contact IBM support if the problem persists.

HSIC085E **GKB schema: *schema* does not have any scorecards.**

section:
The used GKB of the given schema name does not have any scorecards that can be used to match discovered modules. This is most likely to occur if the GKB has not been loaded successfully.

section:
The application terminates.

section:
Ensure GKB is loaded successfully. Contact IBM support if the problem persists.

HSIC086S **SCRT Import detected a configuration error.**

section:
An error occurred while reading the TPARAM dataset.

section:
The application terminates.

section:
Check the log for any additional messages indicating the cause of the error. Contact IBM support.

HSIC087E **SCRT Import detected an input data error.**

section:
The SCRT CSV file is invalid or does not follow the expected format.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC088E **SCRT Import detected a database error.**

section:
An SQL error occurred.

section:
The application terminates.

section:
Check the log for additional information about the error. Contact IBM support.

HSIC089W **Hardware serial *serial* does not exist in the TADzZ SW Asset Mgmt database.**

section:
The hardware serial has not been discovered by Inquisitor Import or Usage Monitor.

section:
The application continues; SCRT data will not be imported for this hardware serial.

section:
Ensure Inquisitor Import and Usage Monitor are run for the desired hardware serial prior to running SCRT Import.

HSIC090W **The following Systems do not exist in the TADzZ SW Asset Mgmt database: *systems***

section:
The systems have not been discovered by Inquisitor Import or Usage Monitor.

section:
The application continues; SCRT data will not be imported for the systems.

section:
Ensure Inquisitor Import and Usage Monitor are run for the desired systems prior to running SCRT Import.

HSIC091W **The following Products do not exist in the current GKB: *products***

section:
The products could not be located in the current GKB.

section:
The application continues; SCRT data will not be imported for the products.

section:
Contact IBM support, supply the missing products information from the log, and request a GKB refresh.

HSIC092W **Node with plant: *plant* serial: *serial* type: *type* does not exist in the FZVSAM database.**

section

No node with the plant, serial, and type provided by SCRT data has been discovered by Inquisitor Import or Usage Monitor.

section

The application continues; SCRT data will not be imported for the specified node.

section

Ensure Inquisitor Import and Usage Monitor are run for the desired nodes prior to running SCRT Import.

Error 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**
6066	**Unknown SID parameter value**
6067	**SQL error**
6068	**Expected parameter missing from the TPARAM table**
6069	**Specified SID is not found**
6070	**Invalid data was encountered**
6071	**Usage Import file is duplicate**
6072	**IQ Import file is duplicate or of earlier date**
6073	**File read error**
6074	**GKB level is unsupported**
6075	**Repository level is unsupported**
6076	**GKB is missing configuration parameters**
6077	**GKB is missing scorecards**
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**
6223	**Error encountered when retrieving the inventory ID**
6224	**Error encountered when retrieving the current GKB version**
6225	**Error encountered when retrieving the inventory GKB version**
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**
6417	**GKB table is empty**
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
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
6666	Error when accessing the TLIBSYS table
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
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
7055	TLPAR record edit 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
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
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
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
7698	Source and destination repositories are not the same type
7699	Source and/or Destination Repositories are not the correct category database
7827	Memory error encountered
8100	SMF Scanner error
8200	SCRT unspecified fatal error
8280	Uneven Max Contributor record elements
8281	Max Contributor record is short or empty
8282	Could not open file

HSII - REXX utility messages

These messages are issued by the REXX utility.

Return codes

Table 109. Return codes and their meaning	
Return code	**Description**
0	No errors encountered. All requests processed successfully.
4	Input/Output 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

Table 110. Message suffix codes and associated condition 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=*rc***

section

The HSIICUST program could not read commands from the SYSIN DDname.

In the message text:

rc
return code from EXECIO

section:
The program terminates and takes no actions.

section:
Correct the JCL and provide a SYSIN DD with valid control statements.

section:
None.

section:
HSIICUST

HSII002I **REQUIRED PARAMETER *prm* IS MISSING FROM SYSIN**

section

The HSIICUST program did not find the required parm in the SYSIN supplied by the user.

In the message text:

prm
name of the parm that is missing.

section:
The program terminates and takes no actions.

section:
Correct the SYSIN and supply the required parm.

section:
None.

section:
HSIICUST

HSII003I **THE DEFAULT VALUE "*prm*=*dft*" IS BEING USED.**

section

The submitted HSISCUST Job SYSIN did not contain this parameter and the default value will now be used.

In the message text:

prm
name of the parameter.

dft
supplied default.

section:
The program continues and uses the default as documented in the message.

section:
If the default value is to be overridden supply the parameter value in the HSISCUST Job SYSIN stream then resubmit.

section:
None.

section:
HSIICUST

HSII004I **ALLOCATION OF DATASET *dsn* TO *dd* FAILED, RC=*rc***

section

The HSIICUST program could not allocate the dataset to the ddname.

In the message text:

dsn
name of the dataset that failed allocation.

dd
DD name to be allocated.

rc
return code from the allocate command.

section:
The program terminates and takes no actions.

section:
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.

section:
None.

section:
HSIICUST

HSII005I ***func* FAILED FOR *rsc*, RC=*rc***

section

The HSIICUST program could not perform a required ISPF function because an error occurred during the function execution.

In the message text:

func
name of ISPF function that failed.

rsc
resource that caused the failure.

rc
return code from the ISPF function.

section:
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.

section:
Check that the options specified do not exceed the field length requirements of the various options. If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSIICUST

HSII006I **MODEL DATASET SHSISAMP HAS NOT BEEN FOUND.**

section:
The HSIICUST program could not find the SHSISAMP dataset created during installation.

section:

The program terminates and takes no actions.

section:
Check that the HSI option is correctly specified and that the installation target libraries are available to the customization program.

section:
None.

section:
HSIICUST

HSII007I CUSTOMIZATION TERMINATES ...

section:
The HSIICUST program encountered an error during execution.

section:
The program terminates and takes no further action.

section:
Check the previous message which will identify the component causing the problem.

section:
None.

section:
HSIICUST

HSII008I *prm* PARAMETER VALUE *pval* IS > *plen* CHARACTERS.

section

The HSIICUST program found a parameter value with a length greater than the allowed value for that parm.

In the message text:

prm
name of HSISCUST parameter that failed.

pval
contents of the PARM.

plen
length of Parameter value.

section:
The program terminates and takes no actions.

section:
Check the parameter in question and re-submit the HSISCUST JCL after correcting the length error.

section:
None.

section:
HSIICUST

HSII009I *prm* DATASET *dsn* WAS NOT FOUND ON SYSTEM *systm*

section

The HSIICUST program found a parameter value, which requires a Dataset name. This Dataset name could not be found on the system.

In the message text:

prm
name of HSISCUST parameter that failed.

dsn
dataset name associated with the PARM.

systm
Operating System name.

section:
The program terminates and takes no actions.

section:
Check the parameter in question and re-submit the HSISCUST JCL after correcting the DSN error.

section:
None.

section:
HSIICUST

HSII010I *verb* STATEMENT VERB NOT ONE OF *list*

section

During syntax parsing for a statement the verb found does not match any of the valid verbs for the program.

In the message text:

verb
word that is not a valid verb.

list
list of valid verbs.

section:
The program terminates and takes no actions.

section:
Update the statements to the program to correct the verb in error and supply a correct verb.

section:
None.

section:
HSIIKBT

HSII011I *word* NOT VALID FOR VERB *verb*

section

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:

word
word that is not valid for the statement syntax for a verb.

verb
the verb of the statement that encountered the error.

section:
The program terminates and takes no actions.

section:
Update the statements to the program to correct the statement in error.

section:
None.

section:
HSIIKBT

HSII012I **PARAMETER *prm* IS NOT A VALID HSISCUST PARAMETER**

section
An invalid HSISCUST SYSIN parameter has been found.

In the message text:

prm
parameter that is invalid.

section:
The program terminates.

section:
Remove the invalid parameter from the HSISCUST Job SYSIN then resubmit.

section:
None.

section:
HSIICUST

HSII013I **PARAMETER *prm* HAS A NULL VALUE**

section
The submitted HSISCUST Job SYSIN has encountered a parameter with a NULL value.

In the message text:

prm
name of the parameter that is null.

section:
The program terminates.

section:
Ensure that the HSISCUST parameter has a valid parameter value in the HSISCUST Job SYSIN then resubmit.

section:
None.

section:
HSIICUST

HSII014I **PARAMETER VALUE FOR *prm* HAS UNBALANCED QUOTES, PARAMETER VALUE IS *pval***

section
An HSISCUST parameter contains unbalanced quotes.

In the message text:

prm
name of the parameter with unbalanced quotes.

pval
parameter value.

section:
The program terminates

section:
Ensure that the Parameter value is enclosed within single quotation marks then resubmit the HSISCUST Job.

section:
None.

section:
HSIICUST

HSII015I **DATASET *dsn stat***

section
The HSIICUST program identifies the status of the output datasets that it is going to use.

In the message text:

dsn
name of output dataset.

stat
status of the output dataset.

section:
The HSIICUST program continues processing.

section:
Informational message, no action required.

section:
None.

section:
HSIICUST

HSII016I **UNMATCHED COMMENT DELIMITER IN HSISCUST STATEMENT: *stmnt***

section
The HSIICUST program found an error in the comment delimiters passed from the HSISCUST SYSIN stream.

In the message text:

stmnt
statement where the error occurred.

section:
The program terminates and takes no actions.

section:
Correct the HSISCUST SYSIN statements and provide valid comment delimiters, /* */.

section:
None.

section:
HSIICUST

HSII017I **PARAMETER *prm* MUST HAVE THE 1ST CHARACTER AS A VALUE BETWEEN A AND Z**

section
The HSISCUST Job has encountered a parameter in the SYSIN DD stream where the first character is not alphabetic. This parameter value must start with a value between A and Z.

In the message text:

prm
name of the parameter that has a non alphabetic first character.

section:
The program terminates.

section:
Ensure that the HSISCUST parameter has a value between A and Z, then resubmit the Job.

section:
None.

section:
HSIICUST

HSII018I ***prm* VALUE *pval* MUST BE WITHIN THE VALID RANGE OF *val1* TO *val2***

section
The HSIICUST program encountered a parameter that was outside the valid range of values allowed.

In the message text:

prm
name of the parameter in error.

pval
value of parameter in error.

val1
minimum value of valid range.

val2
maximum value of valid range.

section:
The program terminates and takes no actions.

section:
Correct the parameter in error and rerun the HSISCUST Job.

section:
None.

section:
HSIICUST

HSII019I **DBTYPE MUST BE THE FIRST PARM IN THE SYSIN STREAM. CURRENT VALUE IS *prm***

section
The HSIICUST program found an error in the first parameter passed from the HSISCUST SYSIN stream.

In the message text:

prm
statement where the error occurred.

section:
The program terminates and takes no actions.

section:
Correct the HSISCUST SYSIN statements and ensure that DBTYPE= is the first entry. Comment statements and blank lines may precede the DBTYPE= entry.

section:
None.

section:
HSIICUST

HSII020I **TAILORING PARAMETERS:**

section:
HSIICUST progress message.

section:
The program continues.

section:
This is an informational progress message and no further action is required.

section:
None.

section:
HSIICUST

HSII021I **CREATING POST-INSTALLATION DATASETS:**

section:
HSIICUST progress message.

section:
The program continues.

section:
This is an informational progress message and no further action is required.

section:
None.

section:
HSIICUST

HSII022I APPLYING TAILORING INFORMATION TO POST-INSTALLATION MEMBERS:

section:
HSIICUST progress message.

section:
The program continues.

section:
This is an informational progress message and no further action is required.

section:
None.

section:
HSIICUST

HSII023I POST-INSTALLATION CUSTOMIZATION COMPLETE.

section:
HSIICUST completion message.

section:
The program ends successfully.

section:
This is an informational progress message and no further action is required.

section:
None.

section:
HSIICUST

HSII024I PARAMETER *prm* CONTAINS AN EXTRANEOUS VALUE: *xval*

section

An HSISCUST parameter contains an extraneous value.

In the message text:

prm
name of the parameter with an extraneous value.

xval
the extraneous value(s).

section:
The program terminates

section:
Ensure that the Parameter value is correctly defined and that any comments are enclosed within comment delimiters.

section:
None.

section:
HSIICUST

HSII025I PARAMETER VALUE FOR *prm* MUST BE IN QUOTES, PARAMETER VALUE IS *pval*

section

An HSISCUST parameter value contains no quotes.

In the message text:

prm
name of the parameter.

pval
parameter value.

section:
The program terminates

section:
Ensure that the parameter value is enclosed within single quotation marks then resubmit the HSISCUST Job.

section:
None.

section:
HSIICUST

HSII026I PARAMETER *prm* MUST BE A VALUE BETWEEN A-Z OR 0-9

section

The HSISCUST Job has encountered a parameter in the SYSIN DD stream where the 1st character is not alphanumeric. This parameter value must be a value between A-Z or 0-9

In the message text:

prm
name of the parameter that has a non alphanumeric first character.

section:
The program terminates.

section:

Ensure that the HSISCUST parameter has a value between A-Z or 0-9, then resubmit the Job.

section:
None.

section:
HSIICUST

HSII027I ***prm* VALUE *pval* FAILED. VALUE MUST BE Db2, SQLITE OR REMOTE.**

section
The HSIICUST program encountered an invalid DBTYPE value.

In the message text:

prm
name of the parameter in error.

pval
value of parameter in error.

section:
The program terminates and takes no actions.

section:
Correct the DBTYPE value and rerun the HSISCUST Job.

section:
None.

section:
HSIICUST

HSII028I **PARAMETERS DB AND DBGKB CANNOT HAVE THE SAME DATABASE NAME: *db***

section
The HSIICUST program found that the values of DB and DBGKB are the same.

In the message text:

db
duplicate Database name.

section:
The program terminates and takes no actions.

section:
Correct the HSISCUST SYSIN statements and provide unique values for both DB and DBGKB.

section:
None.

section:
HSIICUST

HSII100I ***prm* MISSING FROM CONFIGURATION.**

section
The utility requires the parameter in the TPARAM/SYSIN stream.

In the message text:

prm
parameter that is missing.

section:
The program terminates and takes no actions.

section:
Update the parameters in the TPARAM/SYSIN DD to add the missing parameter.

section:
None.

section:
HSIIKBT

HSII107I ***svc* FROM *rsc* FAILED, RC=*rc***

section
An error has occurred executing the service for the resource specified. The service issued the return code mentioned.

In the message text:

svc
service that failed.

rsc
resource that failed.

rc
return code from service.

section:
The program stops processing statements. No changes have been made.

section:
Report this error to the System Programmer.

section:
For "EXECIO READ" service, this means that the resource specified (ddname) is missing or empty.
If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
HSIIKBT

HSII108I **SQL *verb* FAILED, SQLCODE=*sqlcode***

section
An error has occurred executing the SQL verb for the table specified.

In the message text:

verb
SQL verb and table name.

sqlcode
SQLCODE from failing statement.

section:
The program stops processing statements. The current statement changes to Db2 tables are backed out.

section:
Report this error to the system programmer.

section:
If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
HSIIKBT

HSII300I ERROR WRITING TO *ddn*.

section
The XML Export utility has a problem writing the XML file.

In the message text:

ddn
DDNAME of the file

section:
The program terminates.

section:
Check the return code and any preceding messages.

section:
None.

section:
HSIIKBT

HSII301I NUMBER OF LINES WRITTEN TO SWKBTXML DD IS *ocnt*.

section
Number of lines written to SWKBTXML DD by the XML Export utility.

In the message text:

ocnt
lines written to SWKBTXML DD.

section:
The program continues and takes no actions.

section:
Informational message, no action required.

section:
None.

section:
HSIIKBT

HSII302I SQL WARNING FOR *warn*.

section
SQL warning was issued from a command.

In the message text:

warn
SQL warning.

section:
The program continues.

section:
None.

section:
None.

section:
HSIIKBT

HSII303I SQL ERROR FOR *err*.

section
The XML Export Utility has encountered an error.

In the message text:

err
SQL Error.

section:
The program terminates and takes no actions.

section:
Examine the SQL return code to determine the cause of the error. Inform the system programmer.

section:
If you cannot resolve this issue, then gather appropriate diagnostic materials and contact IBM support.

section:
HSIIKBT

HSII304I *err*.

section
The XML Export Utility has encountered a DSNREXX error.

In the message text:

err
DSNREXX error.

section:
The program terminates and takes no actions.

section:
Examine the preceding error messages to determine the error. Inform the system programmer.

section:

If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
HSIIKBT

HSII305I INVALID SCHEMA *sch*.

section
The XML Export Utility has encountered a problem with an invalid schema.

In the message text:

sch
schema name

section:
The program terminates and takes no actions.

section:
Ensure that the correct Schema is being used.

section:
None.

section:
HSIIKBT

HSII999U MODULE HSIMSG FAILED - MSGID=*msgid* RC=*rc* RS=*rs*

section
HSIMSG was called to produce a message text, but the call failed.

In the message text:

msgid
identifier of the failing message.

rc
HSIMSG return code.

rs
HSIMSG reason code.

section:
Terminates with a condition code of 20.

section:
Inform the system programmer.

section:
Contact IBM support.

section:
HSIIINQ

HSIP - Inquisitor for z/OS messages and codes

These messages are issued by the Inquisitor.

Return codes

Table 111. Return codes and their meaning

Return code	Description
0	No errors encountered. All requests processed successfully.
4	Input/Output error in one or more program libraries.
8	Error - Data collection is incomplete. Processing continues. The error is in a system service such as OPEN or DYNALLOC. Data that is collected can be processed normally.
12	Syntax error.
16	Unrecoverable error. No requests processed. Unusable file or unsupported Operating System.
20	Disastrous error. No requests processed. SYSPRINT file cannot be used.

Message suffix codes

Table 112. Message suffix codes and associated condition codes

Suffix	Meaning	Raises minimum condition code to:
I	Information message	0

Table 112. Message suffix codes and associated condition codes (continued)

Suffix	Meaning	Raises minimum condition code to:
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 NO USABLE SYSPRINT FILE

section:
The OPEN of the SYSPRINT file failed. Note: This message is issued by WTO with ROUTCDE=(2,11). All other messages are written to the SYSPRINT file.

section:
Terminates with a condition code of 20.

section:
Ensure a usable SYSPRINT file is allocated. The program overrides any unacceptable DCB values.

section:
None.

section:
HSIPINQ

HSIP001U CANNOT OPEN SYSIN FILE

section:
The OPEN of the SYSIN file failed.

section:
Terminates with a condition code of 16.

section:
Ensure a usable SYSIN file is allocated.

section:
None.

section:
HSIPINQ

HSIP004S UNKNOWN VERB "*verb*"

section

Parsing detected unrecognized data when looking for a verb.

In the message text:

verb
name of the encountered verb.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP005S UNKNOWN OPERAND "*op*"

section

Parsing detected unrecognized data when looking for an operand.

In the message text:

op
name of the encountered operand.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP006S UNEXPECTED OPEN PARENTHESIS ENCOUNTERED

section:

Parsing detected an open parenthesis at an unexpected location.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP007S UNEXPECTED CLOSE PARENTHESIS ENCOUNTERED

section:
Parsing detected a close parenthesis at an unexpected location.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP008S EXPECTED OPEN PARENTHESIS MISSING

section:
Parsing did not detect the expected open parenthesis.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP009S EXPECTED CLOSE PARENTHESIS MISSING

section:
Parsing did not detect the expected close parenthesis.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP010U OPERATING SYSTEM NOT SUPPORTED - CODE "*code*"

section
The value of the byte at CVTDCB was not X'9B'.

In the message text:

code
hexadecimal value of first byte of CVTDCB.

section:
Terminates with a condition code of 16.

section:
This version of the Inquisitor cannot be run on this Operating System. If necessary, gather appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSIPINQ

HSIP011I MISSING CLOSE PARENTHESIS ASSUMED

section:
End-of-file was detected for SYSIN before an expected close parenthesis was detected.

section:
The request is accepted and processing continues.

section:
Correct the SYSIN file contents to avoid this message.

section:
None.

section:
HSIPINQ

HSIP012S MISSING OPERAND SUBPARAMETER FOR *spm*

section
A required subparameter of an operand was not specified.

In the message text:

spm
name of the operand being processed.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:

HSIPINQ

HSIP013S E-O-F INSTEAD OF EXPECTED CONTINUATION

section:
End-of-file was detected for SYSIN instead of an expected record required to continue the current statement being parsed.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP014I COMPLETED REQUEST NUMBER *rno* - PROCESSING STATISTICS ARE:

section
Processing of a request has been completed. One or more messages follow containing the statistics for the request.

In the message text:

rno
sequence number of the request.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIPINQ

HSIP015I VOLUMES=*vol* DATASETS=*ds* BAD-D/S=*dsbad* PROGRAMS=*pgms*

section
Processing of a request has been completed. Statistics related to the request are shown.

In the message text:

vol
count of volumes scanned for this request.

ds
count of data sets successfully processed.

dsbad
count of data sets which could not be processed.

pgms
count of programs processed for this request.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIPINQ

HSIP016I ACCEPTED REQUEST NUMBER *rno*

section
Parsing of a request has been completed successfully. The request is stored for subsequent processing.

In the message text:

rno
sequence number of the request.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP017E DYNALLOC FAILURE: RC=*rc* ERROR=*err* INFO=*inf* VOLUME=*vol*

section
A data set could not be dynamically allocated. See message "HSIP080I" on page 258 for the name of the dataset that incurred the problem.

In the message text:

rc
return code of the DYNALLOC macro.

err
dynamic allocation return code (DARC).

inf
dynamic allocation information code.

vol
volume serial number of the data set.

section:
Processing of this data set is terminated.

section:
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.

section:
None.

section:
HSIPINQ

HSIP018W VTOC DYNALLOC FAILURE: RC=*rc* ERROR=*err* INFO=*inf* VOLUME=*vol*

section
A VTOC could not be dynamically allocated.

In the message text:

rc
return code of the DYNALLOC

err
dynamic allocation return code (DARC).

inf
dynamic allocation information code.

vol
volume serial number of the data set.

section:
Processing of this volume is terminated.

section:
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.

section:
None.

section:
HSIPINQ

HSIP020I *ocnt* INQUISITOR OUTPUT RECORDS WRITTEN

section
Processing has concluded and all data files have been closed.

In the message text:

ocnt
number of records written.

section:
Termination continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP021S INVALID OPERAND SUBPARAMETER FOR *spm*

section
The specified subparameter of an operand was not valid.

In the message text:

spm
name of the operand being processed.

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP022W I/O ERR MEMBER *mbr* IN *dsn*

section
An I/O error was encountered while reading the contents of a load module.

In the message text:

mbr
name of the program being processed.

dsn
name of the data set being processed.

section:
Processing of this member continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP023W ABEND *abend* IN OPEN FOR *dsn*

section
An abnormal end occurred while opening a data set.

In the message text:

abend
hexadecimal system abend and reason

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
None required, but you may wish to exclude the data set from processing, or correct the cause of the abend.

section:
None.

section:
HSIPINQ

HSIP024S BAD UCBSCAN RETURN CODE OF HEX *rc*

section

An unexpected return code was received from UCBSCAN.

In the message text:

rc
hexadecimal return code from UCBSCAN

section:
Processing of volume scanning for this request is terminated.

section:
Rerun the program when no dynamic reconfiguration changes are being implemented.

section:
None.

section:
HSIPINQ

HSIP025U CANNOT OPEN INQPOUT FILE

section:
The OPEN of the INQPOUT file failed.

section:
Terminates with a condition code of 16.

section:
Ensure that the allocated INQPOUT file is usable, or omit the INQPOUT file in favour of using the INQPZIP file.

section:
None.

section:
HSIPINQ

HSIP026E I/O ERROR ENCOUNTERED READING VTOC OF VOLUME *vol* ON DEVICE *dev*

section

An I/O error was encountered while reading a VTOC.

In the message text:

vol
volume serial number being processed.

dev
device number of the volume.

section:
Processing of this track of the VTOC is terminated.

section:
None required, but you may wish to exclude the volume from processing, or correct the cause of the I/O error.

section:
None.

section:
HSIPINQ

HSIP028U CANNOT OPEN INQPDMP FILE

section:
The OPEN of the INQPDMP file failed after DUMPTEXT was specified.

section:
Terminates with a condition code of 16.

section:
Ensure a usable INQPDMP 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.

section:
None.

section:
HSIPINQ

HSIP029I TEXT-DUMPS=*cnt*

section

Processing of a request with DUMPTEXT specified has completed. This message follows “HSIP015I” on page 246.

In the message text:

cnt
count of load module text blocks written.

section:
Processing continues.

section:
None required. The DUMPTEXT operand should only be specified at the request of IBM support.

section:
None.

section:
HSIPINQ

HSIP030I "DUMPTEXT" OPERAND IGNORED FOR "SCANDIR" VERB

section:
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.

section:
The DUMPTEXT operand is ignored and processing continues.

section:
Remove the DUMPTEXT operand to avoid this message. The DUMPTEXT operand should only be specified at the request of IBM support.

section:
None.

section:
HSIPINQ

HSIP031I BAD SELECTION CRITERIA FOR *dsn*

section

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 "HSIP015I" on page 246 which details the cause.

In the message text:

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
If this data set is a program library which should be processed by the Inquisitor then modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP032I OBTAIN FAILED RC=*rc* VOLUME *vol*

section

The system could not read the VTOC entry for the data set named in the "HSIP033I" on page 249 message which follows this message. This message is only issued when DSNMSG or ALLMSG is specified in the program parameter.

In the message text:

rc
hexadecimal return code of the OBTAIN macro.

vol
volume serial number being processed.

section:
Processing of this data set is terminated.

section:
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. If necessary, gather appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSIPINQ

HSIP033I OBTAIN FAILED FOR DATA SET *dsn*

section

The system could not read the VTOC entry for the data set on the volume named in the previous "HSIP032I" on page 249 message. This message is only issued when DSNMSG or ALLMSG is specified in the program parameter.

In the message text:

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
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. If necessary, gather appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSIPINQ

HSIP034I REFER DATE WAS *date* FOR *dsn*

section

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 DSNMSG or ALLMSG is specified in the program parameter.

In the message text:

date
the Julian reference date from the VTOC entry.

dsn
name of the data set being processed.

section:
Processing of this data set continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP036E **OPEN ERROR ENCOUNTERED READING VTOC OF VOLUME *vol* ON DEVICE *dev***

section

The VTOC of the volume shown could not be opened.

In the message text:

vol
volume serial number being processed.

dev
device number of the volume.

section:
Processing of this track of the VTOC is terminated.

section:
None required, but you may wish to exclude the volume from processing, or correct the cause of the I/O error.

section:
None.

section:
HSIPINQ

HSIP037E **SECURITY ACCESS DENIED FOR *dsn***

section

A RACROUTE macro determined the program had insufficient security access to read the data set.

In the message text:

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
Contact Security Administration to obtain sufficient security access to read the data set or exclude the data set from processing.

section:
None.

section:
HSIPINQ

HSIP038I **BAD SELECTION CRITERIA WAS *dsn***

section

Processing of a data set was specified but attributes did not match other selection criteria also specified in the request. This message follows “HSIP031I” on page 249 which shows the data set name.

In the message text:

dsn
cause of the data set processing failure.

section:
Processing of this data set is terminated.

section:
If this data set is a program library which should be processed by the Inquisitor then modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP039S **ALL POSSIBLE DATA SETS ARE EXCLUDED**

section:
An exclusion mask has been specified which excludes all possible data sets included by a selection mask. Both masks are shown after this message.

section:
Terminates with a condition code of 12.

section:
Modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP040S **ALL POSSIBLE DASD VOLUMES ARE EXCLUDED**

section:
An exclusion mask has been specified which excludes all possible DASD volumes included by a selection mask. Both masks are shown after this message.

section:
Terminates with a condition code of 12.

section:
Modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP041S **ALL POSSIBLE PROGRAMS ARE EXCLUDED**

section:
An exclusion mask has been specified which excludes all possible programs included by a selection mask. Both masks are shown after this message.

section:
Terminates with a condition code of 12.

section:
Modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP042S ALL POSSIBLE MODULES ARE EXCLUDED

section:
An exclusion mask has been specified which excludes all possible modules included by a selection mask.

section:
Terminates with a condition code of 12.

section:
Modify or remove the conflicting selection criteria. If no CSECT-level records are required then omit both MODULE and XMODULE operands.

section:
None.

section:
HSIPINQ

HSIP043I "MODULE"/"CSECT" OPERAND IGNORED FOR "SCANDIR" VERB

section:
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.

section:
The MODULE operand is ignored and processing continues.

section:
Remove the MODULE operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP044I "XMODULE"/"XCSECT" OPERAND IGNORED FOR "SCANDIR" VERB

section:
An XMODULE operand was encountered for a SCANDIR request. That is, the output of program structure data was implied in a request which does not have access to this data.

section:
The XMODULE operand is ignored and processing continues.

section:
Remove the XMODULE operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP045I THE "XDSNAME" MASK IS NOT A SUBSET OF ANY "DSNAME" MASK

section:
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.

section:
Processing continues.

section:
Specify the XDSNAME operand as a further qualification of the DSNAME operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP046I THE "XVOLUME" MASK IS NOT A SUBSET OF ANY "VOLUME" MASK

section:
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.

section:
Processing continues.

section:
Specify the XVOLUME operand as a further qualification of the VOLUME operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP047I THE "XPROGRAM" MASK IS NOT A SUBSET OF ANY "PROGRAM" MASK

section:

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.

section:
Processing continues.

section:
Specify the XPROGRAM operand as a further qualification of the PROGRAM operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP048I THE "XMODULE" MASK IS NOT A SUBSET OF ANY "MODULE" MASK

section:
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.

section:
Processing continues.

section:
Specify the XMODULE operand as a further qualification of the MODULE operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP049I THE "XSTOGROUP" MASK IS NOT A SUBSET OF ANY "STOGROUP" MASK

section:
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.

section:
Processing continues.

section:
Specify the XSTOGROUP operand as a further qualification of the STOGROUP operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP050I MODULES=*cnt*

section
Processing of a request with MODULE specified has completed. This message follows "HSIP015I" on page 246.

In the message text:

cnt
count of CSECTs processed in this request.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP051I ****** PARSE ONLY REQUEST PROCESSED - NO ACTION TAKEN **********

section:
Processing of a SCANCMD request has completed.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP052U MISSING INQPOUT AND INQPZIP FILES

section:
Neither an INQPOUT nor an INQPZIP file is allocated. At least one output file is required.

section:
Terminates with a condition code of 16.

section:
Specify an output file and rerun the job.

section:
None.

section:
HSIPINQ

HSIP053U COMPRESSION SUBROUTINE ERROR

section:

While processing the INQPZIP file the compression subroutine encountered an error. The error message from the compression subroutine immediately follows this message.

section:
Terminates with a condition code of 16.

section:
Correct the error described in the message from the compression subroutine. If necessary, gather appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSIPINQ

HSIP054I **"FULLIDR" OPERAND IGNORED FOR "SCANDIR" VERB**

section:
A FULLIDR operand was encountered for a SCANDIR request. That is, the processing of load module member contents was specified in a request which does not have access to this data.

section:
The FULLIDR operand is ignored and processing continues.

section:
Remove the FULLIDR operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP056I ***date time*** **COMMENCING SCAN OF VOLUME** ***vol*** **ON UNIT** ***unit***

section

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:

date
date of message.

time
time of message.

vol
serial number of volume.

unit
device number of volume.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP057E **ABEND** ***abend*** **PROCESSING VTOC OF VOLUME** ***vol*** **ON UNIT** ***unit***

section

A request without the CATALOG keyword attempted to process a DASD volume VTOC and the OPEN or CLOSE abended. The volume may not be usable.

In the message text:

abend
hexadecimal system abend and reason codes.

vol
serial number of volume.

unit
device number of volume being processed.

section:
Processing of this volume is terminated.

section:
Vary the volume offline, and/or reformat the volume. Institute any appropriate volume recovery procedures.

section:
None.

section:
HSIPINQ

HSIP058S **DUPLICATE OPERAND ENCOUNTERED:** ***op***

section

An input request was found to have the indicated operand specified more than once.

In the message text:

op
name of the duplicate operand

section:
Terminates with a condition code of 12.

section:
Correct the SYSIN file contents and rerun the program.

section:
None.

section:
HSIPINQ

HSIP059W **BINDER FAILURE FOR MEMBER *mbr* RC=*rc* RS=*rs***

section

The Binder could not successfully process a member of a PDSE.

In the message text:

mbr
name of the member being processed.

rc
hexadecimal Binder FDA API return code.

rs
hexadecimal Binder FDA API reason code.

section:
Terminates data collection for this member, writes out data already collected and continues processing the next member.

section:
None required.

section:
The Binder Fast Data Access API return and reason codes provide more detailed indication of the cause.

section:
HSIPINQ

HSIP060S **SYMBOL SUBSTITUTION FAILURE - ASASYMBP RC=*rc***

section

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:

rc
hexadecimal return code.

section:
Terminates with a condition code of 12.

section:
Correct or remove the symbols in control statement input.

section:
None.

section:
HSIPINQ

HSIP061I ***pgm* NON-REEDITABLE IN *dsn***

section

A program object in a PDSE was encountered which cannot be processed by the Program Binder. The program was bound with the NE or OVLY attribute. This message is only issued when PGMMSG or ALLMSG is specified in the program parameter. This message is not issues for DLL-enable program objects.

In the message text:

pgm
name of program which cannot be processed.

dsn
name of the data set being processed.

section:
Further data collection for this member is terminated.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP062S **THE CATALOG REQUEST NEEDS EXACTLY ONE DSNAME MASK**

section:
A request with the CATALOG operand either omitted the DSNAME operand or specified more than one DSNAME mask.

section:
Terminates with a condition code of 12.

section:
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.

section:
None.

section:
HSIPINQ

HSIP063S **ALL POSSIBLE STORAGE GROUPS ARE EXCLUDED**

section:
An exclusion mask has been specified which excludes all possible storage groups included by the selection mask. Both masks are shown after this message.

section:
Terminates with a condition code of 12.

section:
Modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP064W ABEND *abend* FOR *mbr* IN *dsn*

section
A subtask processing a program object from a PDSE has abended. The abend probably occurred in the Program Binder API.

In the message text:

abend
hexadecimal system abend code.

mbr
name of the member being processed.

dsn
name of the data set being processed.

section:
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.

section:
Exclude the programs causing the failure and rerun the Job.

section:
None.

section:
HSIPINQ

HSIP065S MCDS FILE FAILED VERIFICATION

section:
The MCDS data definition (DD) was found to be unusable by the Inquisitor. One or more of the following is true: 1) The MCDS file could not be opened. Message "HSIP066E" on page 255 follows. 2) The MCDS file is not a VSAM key-sequenced data set (KSDS). 3) The KSDS relative key position (RKP) is not zero (0). 4) The KSDS key length is not forty-four (44).

section:
Terminates with a condition code of 12.

section:
Either ensure that the Inquisitor has read access to DFHSM's MCDS, or change 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.

section:
None.

section:
HSIPINQ

HSIP066E MCDS OPEN ERROR - RC=*rc* RS=*rs*

section
The OPEN of the MCDS data definition (DD) was not successful.

In the message text:

rc
VSAM OPEN hexadecimal return code.

rs
VSAM OPEN hexadecimal reason code.

section:
Issues message "HSIP065S" on page 255 and terminates with a condition code of 12.

section:
Either ensure that the Inquisitor has read access to DFHSM's MCDS, or modify 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.

section:
None.

section:
HSIPINQ

HSIP067E MCDS READ RC=*rc* RS=*rs* FOR *dsn*

section
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:

rc
VSAM GET hexadecimal return code.

rs
VSAM GET hexadecimal reason code.

dsn
name of data set cataloged on volume MIGRAT.

section:
Processing of this data set is terminated.

section:
If the data set is not really migrated then correct the catalog entry. If the MCDS is corrupt then begin recovery procedures.

section:
None.

section:
HSIPINQ

HSIP068W CATALOG RC=*rc* RS=*rs*,*modid cat*

section

The Catalog Search Interface returned an entry which is flagged as being in error by Catalog Management.

In the message text:

rc
Catalog Management decimal return code.

rs
Catalog Management decimal reason code.

modid
Catalog Management module identifier.

cat
name of catalog entry in error.

section:
Processing of this data set is terminated.

section:
Correct the catalog entry. Refer to the System Messages manual for message IDC3009I to find out the meaning of the Catalog Management error codes.

section:
None.

section:
HSIPINQ

HSIP069U **PROGRAM IS NOT APF AUTHORIZED**

section:
The Inquisitor has determined that it is not running in an APF authorized environment, and PARM=NOAPF was not specified.

section:
Terminates with a condition code of 20.

section:
Ensure that the HSIPINQ program is run in an APF authorized environment, or specify PARM=NOAPF in the JCL.

section:
None.

section:
HSIPINQ

HSIP070E **BAD BLKSIZE AFTER OPEN FOR *dsn***

section

A BPAM DCB was opened for the named PDS, but despite the VTOC entry indicating a suitable blocksize, the blocksize in the DCB after the OPEN was not positive.

In the message text:

dsn
name of the data set being processed.

section:
Processing of member contents for this data set is terminated to avoid an S002-30 abend.

section:
The PDS is probably corrupt and should be deleted. Recreate it from a backup if appropriate.

section:
None.

section:
HSIPINQ

HSIP071W **IGNORING INVALID DSNAME IN *dsn***

section

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:

dsn
name of the catalog being processed.

section:
The returned catalog entry is discarded.

section:
Ensure that the named catalog is not corrupt and contains no invalid entries.

section:
None.

section:
HSIPINQ

HSIP072I **BYPASS PROCESSING DATA SET *dsn* DUE TO *patn* NAME PATTERN MATCH**

section

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. The matching name pattern will be one of DLIB (distribution library), RELFILE (SMP/E relative file), SMPLTS (SMP/E Load Temporary Store library) or IMS LIBRARY. This message is only issued when DSNMSG or ALLMSG is specified in the program parameter.

In the message text:

dsn
name of the data set being bypassed.

patn
matching data set type name pattern.

section:
The data set is not opened, and no data from it is collected.

section:
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.

section:
None.

section:
HSIPINQ

HSIP073I NO DATA WAS EXTRACTED FROM *dsn*

section
The data set contained no members eligible for selection. This message is only issued when DSNMSG or ALLMSG is specified in the program parameter.

In the message text:

dsn
name of the processed data set.

section:
The data set was opened, but no data from it is collected.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP074S ABRIN OR ABRPRINT FILES NOT ALLOCATED

section:
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.

section:
Terminates with a condition code of 12.

section:
Ensure the required files are pre-allocated for the Inquisitor.

section:
None.

section:
HSIPINQ

HSIP075W FDRABR ABEND *abend* CHECKING *dsn*

section
An abend occurred during ABR processing while checking a data set which may have been archived.

In the message text:

abend
hexadecimal system abend code.

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
Ensure the catalog entry for the data set is correct.

section:
None.

section:
HSIPINQ

HSIP076E BAD LOAD *abend-rs*: *mbr dsn*

section
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:

abend
abend code returned by LOAD.

rs
abend reason code returned by LOAD.

mbr
name of the member containing the tag data.

dsn
name of the data set containing the tag data module.

section:
Processing continues with the next member in the data set.

section:
Verify that the named data set contains no unusable modules. If necessary, delete any modules that are of no further use.

section:
None.

section:
HSIPINQ

HSIP077W ISITMGD RC=*rc* RS=*rs* FOR *dsn*

section
The Inquisitor executed an ISITMGD macro for the named data set, but ISITMGD issued a non-zero return code.

In the message text:

rc
decimal return code issued by ISITMGD.

rs
hexadecimal reason code issued by ISITMGD.

dsn
name of the data set being processed.

section:
Processing continues with the next data set.

section:
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.

section:
None.

section:
HSIPINQ

HSIP078W DESERV RC=*rc* RS=*rs* FOR *dsn*

section
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:

rc
decimal return code issued by DESERV.

rs
decimal reason code issued by DESERV.

dsn
name of the data set being processed.

section:
Processing continues with the next data set.

section:
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.

section:
None.

section:
HSIPINQ

HSIP080I DYNALLOC FAILURE: DSN=*dsn*

section
A data set could not be dynamically allocated.

In the message text:

dsn
name of the data set being processed.

section:
Depends upon other messages associated with this message.

section:
None required.

section:
None.

section:
HSIPINQ

HSIP081S ALL POSSIBLE DEVICE NUMBERS ARE EXCLUDED

section:
An exclusion mask has been specified which excludes all possible device numbers included by a selection mask. Both masks are shown after this message.

section:
Terminates with a condition code of 12.

section:
Modify or remove the conflicting selection criteria.

section:
None.

section:
HSIPINQ

HSIP082I THE "XDEVICE" MASK IS NOT A SUBSET OF ANY "DEVICE" MASK

section:
The mask specified in the XDEVICE operand excludes possible values not included in the DEVICE operand. This message is issued to highlight possible inconsistencies in a request.

section:
Processing continues.

section:
Specify the XDEVICE operand as a further qualification of the DEVICE operand to avoid this message.

section:
None.

section:
HSIPINQ

HSIP083E **RENAME FAILED FOR DATA SET *dsn***

section

The rename operation to add one or more extra low-level qualifiers to a data set name as specified by the LLQ program parameter setting did not succeed. The named data set is allocated to either the INQPZIP or INQPOUT file. This message is preceded by either an associated explanatory message, or by messages from IDCAMS detailing the results of the rename attempt.

In the message text:

dsn
name of the INQPZIP or INQPOUT data set.

section:
The output data set retains its original name.

section:
Ensure that the specified LLQ string length does not exceed 44 bytes, that any symbols used are valid for this system, and that resultant data set names are not longer than 44 bytes. Examine associated messages to determine the reason for the rename failure.

section:
None.

section:
HSIPINQ

HSIP084I **ABEND *abend* OPENING DSN *dsn***

section

An abnormal end occurred while opening a data set.

In the message text:

abend
hexadecimal system abend and reason

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
None required, but you may wish to correct the cause of the abend.

section:
None.

section:
HSIPINQ

HSIP085I **ABEND *abend* CLOSING DSN *dsn***

section

An abnormal end occurred while closing a data set.

In the message text:

abend
hexadecimal system abend and reason

dsn
name of the data set being processed.

section:
Processing of this data set is terminated.

section:
None required, but you may wish to correct the cause of the abend.

section:
None.

section:
HSIPINQ

HSIP086S **NO PROGRAMS OR TAG DATA FOUND - NO DATA FOR IMPORT WAS PRODUCED**

section:
All scanning operations failed to find any executable programs or program tag data, so no data suitable for subsequent processing was created.

section:
Terminates with a condition code of 12.

section:
Correct any selection criteria errors and rerun the job.

section:
None.

section:
HSIPINQ

HSIP087I **SKIP INACCESSIBLE DATA SET *dsn***

section

The named data set was encountered on an SMS-managed volume, but no matching catalog entry could be located, which means that the data set cannot be successfully allocated by any job.

In the message text:

dsn
name of the data set being skipped.

section:
The data set is bypassed and processing continues.

section:
None required, but you may wish to either catalog the data set to make it accessible, or delete it to reclaim the disk space.

section:
None.

section:
HSIPINQ

HSIP088I **DEVICES=*vol* CU-GROUPS=*ds* USED-CHPIDS=*dsbad***

section

Processing of a SCANDEV request has been completed. Counts of online I/O devices, device groups and used channel paths are shown. A group of devices with the same device type, control unit, and CHPID connectivity generates one CU record. A CP record is generated for each online CHPID connected to at least one online I/O device.

In the message text:

vol
count of online I/O devices discovered.

ds
count of CU records written.

dsbad
count of CP records written.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIPINQ

HSIP089E **INSUFFICIENT AUTHORIZATION TO PROCESS "SCANDEV" REQUEST**

section:
To process a SCANDEV request either the Inquisitor must be APF authorized or the user must have UPDATE access to the IOSCDR entity in the FACILITY security class.

section:
Processing continues with the next request.

section:
Get the appropriate authorization, or omit the SCANDEV request.

section:
None.

section:
HSIPINQ

HSIP090I **EXCESSIVE DESERV DELAY FOR *dsn***

section

This message is issued when a DESERV macro has not returned control to the Inquisitor after 15 minutes. This might indicate that the data set being processed is corrupt and unusable.

In the message text:

dsn
name of the data set being scanned.

section:
Processing continues, possibly without making any further progress.

section:
If the scan does not progress, and the delay is not caused by contention with other work, cancel the job. Either delete the data set, or add an exclusion to the Inquisitor control statement before rerunning the job.

section:
None.

section:
HSIPINQ

HSIP091I **IGNORING OLD TAG MEMBER mem IN DATA SET=*dsn***

section

This message is issued when a tag member made by the defunct Tagger program HSITAGP is encountered. The Inquisitor will not collect any information from or about this member.

In the message text:

mem
name of the found tag member.

section

In the message text:

dns
name of the data set being scanned.

section:
Processing continues.

section:
The tag member can be safely deleted. Local tagging functions are now available in the Analyzer which can directly update the data base.

HSIP092I **ANOTHER JOB HOLDS dsn =*time / dsn***

section

This message is issued when a library to be scanned is found to be exclusively allocated to another job, and therefore cannot be scanned.

In the message text:

time
time of message.

dns
name of the unavailable data set.

section:
Processing continues. If the data set does not become available before the processing of the request completes, an error or warning condition will be raised. The default error condition can be reduced to a warning condition by specifying the INUSEWARN keyword in the request statement.

section:
If necessary, and if message "HSIP093I" on page 261 is not issued for this data set, rerun the scan when the data set is available.

HSIP093I **RETRY SUCCESS FOR vol / dsn =*vol / dsn***

section
This message is issued when a library to be scanned was previously found to be exclusively allocated to another job, but has now become available and has been processed.

In the message text:

vol
serial number of volume.

dns
name of the unavailable data set.

section:
Processing continues.

section:
None.

HSIP094W **RETRY FAILURE FOR vol / dsn =*vol / dsn***

section
This message is issued when a library to be scanned was previously found to be exclusively allocated to another job, and the data set retried. INUSEWARN was specified on the request.

In the message text:

vol
serial number of volume.

dns
name of the unavailable data set.

section:
Processing continues.

section:
If necessary, rerun the scan when the data set is available.

HSIP095E **RETRY FAILURE FOR vol / dsn =*vol / dsn***

section
This message is issued when a library to be scanned was previously found to be exclusively allocated to another job, and the data set remains unavailable when the allocation was retried. INUSEWARN was not specified on the request.

In the message text:

vol
serial number of volume.

dns
name of the unavailable data set.

section:
Processing continues.

section:
If necessary, rerun the scan when the data set is available.

HSIP097E **CATALOG SEARCH INTERFACE ERROR RC=*csirc***

section
A request with the CATALOG keyword was specified, and the Catalog Search Interface encountered an error.

In the message text:

csirc
return code from the Catalog Search Interface.

section:
Processing catalog entries for the request is terminated.

section:
Correct any related catalog errors.

section:
None.

section:
HSIPINQ

HSIP098E **CATALOG SEARCH INTERFACE ERROR RC=*csirc* CATALOG RC=*rc* CATALOG RS=*rs***

section
A request with the CATALOG keyword was specified, and the Catalog Search Interface encountered an error.

In the message text:

csirc
return code from the Catalog Search Interface.

rc
return code from Catalog Management.

rs
reason code from Catalog Management.

section:
Processing catalog entries for the request is terminated.

section:
Correct any related catalog errors.

section:
None.

section:
HSIPINQ

HSIP099E **CATALOG SEARCH INTERFACE ERROR RC=*csirc* CATALOG RC=*rc* CATALOG RS=*rs* MODULE=*modid***

section

A request with the CATALOG keyword was specified, and the Catalog Search Interface encountered an error.

In the message text:

csirc
return code from the Catalog Search Interface.

rc
return code from Catalog Management.

rs
reason code from Catalog Management.

modid
module identifier.

section:
Processing catalog entries for the request is terminated.

section:
Correct any related catalog errors.

section:
None.

section:
HSIPINQ

HSIP100I **PROGRAM HSIPVMAP COMPLETED WITH RC=*rc***

section

The HSIPVMAP program was invoked to process the optional DASDMAP file, and HSIPVMAP issued the return code shown. A return code of zero indicates success. A return code of 8 indicates that the DASDMAP DD was allocated, but a problem was encountered trying to process it.

In the message text:

rc
decimal return code from HSIPVMAP.

section:
Processing continues.

section:
If necessary, investigate any I/O errors associated with the data set allocated to DASDMAP. Reconcile the intended mapping(s) with any HSIP101I messages that immediately follow. If necessary, correct the assignments and rerun the job.

section:
None.

section:
HSIPINQ

HSIP101I **VOLUMES MATCHING *"v-mask"* WILL BE MAPPED TO *"v-name"***

section

The HSIPVMAP program has extracted the displayed logical volume mapping from the DASDMAP file.

In the message text:

v-mask
volume serial selection mask.

v-name
assigned logical volume name.

section:
Processing continues.

section:
Verify that the displayed logical volume mapping is as intended. Note that while the Usage Monitor also calls program HSIPVMAP, it does not externalize the results, so trial executions of the HSIPINQ program should be used to test new logical volume mapping assignments before the resulting data is imported into the data base.

section:
None.

section:
HSIPINQ

HSIP102U **DSPSERV CREATE RC=*rc* RS=*rs***

section

The Inquisitor executed a DSPSERV CREATE macro to acquire storage to hold collected data, but DSPSERV issued a non-zero return code.

In the message text:

rc

hexadecimal DSPSERV return code.

rs

hexadecimal DSPSERV reason code.

section

Program HSIPINQ abends with user abend code 29 and with the reason code returned by DSPSERV.

section

Consult the applicable MVS Programming: Assembler Services Reference manual to determine the meaning of the DSPSERV return and reason codes. Ensure that the job is permitted to acquire numerous data spaces and gigabytes of data space storage.

section

Inspect and, if necessary, alter the DSLIMITNUM and DSLIMITSIZE settings in the SMFLIMxx PARMLIB member that are applied to this job.

section:
HSIPINQ

HSIP103I ***type*** **QUEUE MAXIMUM=*count***

section

This message is issued when volume-scanning subtasks queue collected data about items such as data sets and programs for processing by the main task. The maximum queue length for each item type is reported.

In the message text:

type

item type - refer to message HSIP104I for possible values.

count

maximum queue length.

section

Processing continues.

section

None.

section

None.

section:
HSIPINQ

HSIP104I ***type*** **DATA SPACE FREE STORAGE ADDRESS IS *addr***

section

This message is issued when volume-scanning subtasks use data space storage to queue collected data for processing by the main task. The start address of unused virtual storage in the relevant data space is reported. Pages above this address were not referenced. This message indicates how much data space storage was used by the request. The used storage is also reported as a percentage of the total data space size to provide capacity feedback.

In the message text:

type

item type. The following data categories might be reported:

- DATASET - contains details of data sets selected for scanning.
- PROGRAM - contains details of programs scanned by subtasks.
- PGMTEXT - contains eyecatchers and character strings from programs scanned by subtasks.
- SECTION - contains details of control sections and LE compile units extracted from programs scanned by subtasks. The data space to hold this item type is only created for SCANPGM FULLIDR requests.

addr

8-digit hexadecimal virtual storage address followed by a percentage in parentheses.

section

Processing continues.

section

None.

section

None.

section:
HSIPINQ

HSIP105I **USED DATA SPACE PAGES:*size***

section

This message is issued when volume-scanning subtasks use data space storage to queue collected data for processing by the main task. It reports the maximum amount of virtual storage across all data spaces created by the main task for this request that needed to be backed in central and/or auxiliary storage by the system. This storage quantity is also expressed as a percentage to indicate how much of the data space storage that SMF may report as being allocated was actually used. The small data space that may be created by each volume-scanning subtask is not included in these numbers.

In the message text:

size

virtual storage size, followed by a percentage in parentheses.

section

Processing continues.

section

None.

section

None.

section:
HSIPINQ

HSIP106I ***bytes*** **COMPRESSED DATA BYTES WRITTEN TO** ***ddname***

section

This message is issued after closing the named output DD which contains compressed (zipped) data. The reported compressed data byte count matches the count contained in the zip file headers, so the count does not include the size of the zip headers. The reported DD name will be INQPZIP unless HSIPINQ was invoked by another program which supplied a DD override parameter.

In the message text:

bytes
byte count.

ddname
output DD name.

section

Processing continues.

section

None.

section

None.

section:
HSIPINQ

HSIP107I **NUMBER OF CONCURRENT DATA SPACES LIMIT IS** ***limit***

section

This message is issued when a relatively low concurrent data space count limit is detected. Ensure that active SMFLIMxx PARMLIB settings do not unduly limit storage access for the job. The Inquisitor uses data space storage to reduce the dependency on large region size availability, and depending on the actual request, may allocate up to MAXTASKS+4 data spaces.

In the message text:

limit

maximum allowed data space count.

section

Processing continues.

section

None.

section

None.

section:
HSIPINQ

HSIP108I **DATA SPACE STORAGE LIMIT IS** ***limit*** **MB**

section

This message is issued when a relatively low data space storage limit is detected. Ensure that active SMFLIMxx PARMLIB settings do not unduly limit storage access for the job. The Inquisitor uses data space storage to reduce the dependency on large region size availability, and depending on the actual request, may allocate up to four 2 GB and MAXTASKS 16648 KB data spaces.

In the message text:

limit

maximum allowed data megabyte count.

section

Processing continues.

section

None.

section

None.

section:
HSIPINQ

HSIP999U **MODULE HSIPMSG FAILED - MSGID=*msgid* RC=*rc* RS=*rs***

section

HSIMSG was called to produce a message text, but the call failed.

In the message text:

msgid
identifier of the failing message.

rc
HSIMSG return code.

rs
HSIMSG reason code.

section:
Terminates with a condition code of 20.

section

Ensure JOBLIB/STEPLIB contains the library where the HSIPMSG message module resides. If you cannot resolve this issue, contact IBM support.

section:
HSIPINQ

HSIT - Product tagging messages

These messages are issued by the Product Tagger.

Return codes

Table 113. Return codes and their meaning

Return code	Description
0	No errors encountered. All requests processed successfully.
4	Warning issued. Processing continues. Input/Output error in one or more program libraries.
8	Error - Incomplete data. Processing continues. OPEN or other system service error.
12	Syntax 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

Table 114. Message suffix codes and associated condition 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

Table 114. Message suffix codes and associated condition codes (continued)		
Suffix	**Meaning**	**Raises minimum condition code to:**
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 *file*

section

A required file could not be opened successfully.

In the message text:

file
name of file.

section:
Processing terminates with condition code 16.

section:
Correct the file definition and rerun the job.

section:
None.

section:
HSITAGP

HSIT002S UNRECOGNIZED STATEMENT TYPE: *stattyp*

section

Input text was encountered which does not match any known statement type.

In the message text:

stattyp
encountered input data.

section:
Processing terminates with condition code 12.

section:
Correct the input and rerun the job.

section:
None.

section:
HSITAGP

HSIT003S DUPLICATE VALUE SUPPLIED FOR *stattyp*

section

More than one occurrence of the named statement type was encountered, but only one value can be accepted.

In the message text:

stattyp
name of the statement verb.

section:
Processing terminates with condition code 12.

section:
Remove the redundant statement and rerun the job.

section:
None.

section:
HSITAGP

HSIT004S VALUE MISSING IN *stattyp* STATEMENT

section

An input statement of the type indicated was encountered, but no non-blanks followed the statement type name.

In the message text:

stattyp
name of the statement verb.

section:
Processing terminates with condition code 12.

section:

Supply an appropriate value after the statement type name.

section:
None.

section:
HSITAGP

HSIT005S VALUE SPECIFIED FOR LICENSED WAS NEITHER "YES" NOR "NO"

section:
A LICENSED statement was processed which had a value specified other than one of the valid values.

section:
Processing terminates with condition code 12.

section:
Correct the value and rerun the job.

section:
None.

section:
HSITAGP

HSIT006S THE *parm* PARAMETER HAD NO SUBPARAMETER VALUE SPECIFIED

section
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:

parm
name of the parameter or operand being processed when the error is detected.

section:
Processing terminates with condition code 12.

section:
Correct the input and rerun the job.

section:
None.

section:
HSITAGP

HSIT007I A CLOSING PARENTHESIS ASSUMED FOR *parm*

section
End-of-file was raised when processing input statements before an expected close parenthesis was encountered.

In the message text:

parm
name of the parameter or operand being processed when the error is detected.

section:
Processing continues as if the expected close parenthesis had been specified.

section:
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.

section:
None.

section:
HSITAGP

HSIT008S UNEXPECTED OPEN PARENTHESIS ENCOUNTERED AFTER *parm*

section
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:

parm
name of the parameter or operand being processed when the error is detected.

section:
Processing terminates with condition code 12.

section:
Correct the input file and rerun the job.

section:
None.

section:
HSITAGP

HSIT009S UNEXPECTED CLOSE PARENTHESIS ENCOUNTERED AFTER *parm*

section
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:

parm
name of the parameter or operand being processed when the error is detected.

section:

Processing terminates with condition code 12.

section:
Correct the input file and rerun the job.

section:
None.

section:
HSITAGP

HSIT010S ***parm*** **IS AN UNKNOWN SELECT PARAMETER**

section
Input data was encountered which is not a recognized parameter, or operand, of the SELECT statement.

In the message text:

parm
the encountered input data.

section:
Processing terminates with condition code 12.

section:
Correct the input file and rerun the job.

section:
None.

section:
HSITAGP

HSIT011S **MEMBER NAME** ***parm*** **HAS EMBEDDED BLANK(S)**

section
The value specified on the TAGMEM statement was not a valid partitioned data set member name, a blank was found within the eight character member name.

In the message text:

parm
the input value specified on the TAGMEM statement.

section:
Processing terminates with condition code 12.

section:
Correct the input file and rerun the job.

section:
None.

section:
HSITAGP

HSIT012S **MISSING OPEN PARENTHESIS AFTER** ***parm***

section
Whilst parsing the SELECT 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:

parm
name of the parameter or operand being processed when the error is detected.

section:
Processing terminates with condition code 12.

section:
Correct the input and rerun the job.

section:
None.

section:
HSITAGP

HSIT013S **VALUE** ***data*** **TOO LONG FOR PARAMETER** ***parm***

section
The length of a subparameter or value was found to exceed the maximum length allowed. The maximum length allowed 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 six characters in length.

In the message text:

data
encountered input data.

parm
name of the parameter or operand being processed when the error is detected.

section:
Processing terminates with condition code 12.

section:
Correct the input and rerun the job.

section:
None.

section:
HSITAGP

HSIT014S **END OF INPUT REACHED, EXPECTED CONTINUATION IS MISSING**

section:

End-of-file was raised on the input (SYSIN) file, but the SELECT statement currently being processed was expected to continue on the next record.

section:
Processing terminates with condition code 12.

section:
Either supply the missing input data or remove the continuation character from the last input record. Rerun the job.

section:
None.

section:
HSITAGP

HSIT017S **NO VALUE FOR *stattyp* WAS SPECIFIED**

section

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:

stattyp
the type of input statement required to specify the missing value.

section:
Processing terminates with condition code 12.

section:
Supply a statement of the named type which specifies a value.

section:
None.

section:
HSITAGP

HSIT019E **DESERV FAILED - RC=*rc* RS=*rs* FOR DATA SET *dsn***

section

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:

rc
the decimal return code issued by DESERV.

rs
the hexadecimal reason code issued by DESERV.

dsn
the name of the data set being processed by DESERV.

section:
The named data set is not processed, and processing continues with the next relevant data set.

section:
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.

section:
None.

section:
HSITAGP

HSIT020S **DYNAMIC ALLOCATION FAILURE - BPXWDYN RC=*rc***

section

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:

rc
the hexadecimal return code issued by BPXWDYN.

section:
Processing terminates with condition code 12.

section:
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.

section:
None.

section:
HSITAGP

HSIT022S **RC=*rc* WAS RETURNED BY PROGRAM *pgm***

section

Either the High Level Assembler (program ASMA90) or the Program Binder (program IEWL) was dynamically started to assist with creating the output data, but the named program issued a non-zero return code.

In the message text:

rc
the decimal return code issued by the named program.

pgm
the name of the program that was started.

section:
Processing terminates with condition code 12.

section:
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.

section:
None.

section:
HSITAGP

HSIT023I **PROCESSING TERMINATED DUE TO ENCOUNTERED ERROR CONDITION**

section:
Because of a previously reported error, the product tagging utility is terminating unilaterally, without processing all of the specified program library data sets, and without generating all of the requested program product tagging data.

section:
Processing terminates.

section:
Investigate any previously reported error conditions.

section:
None.

section:
HSITAGP

HSIT024E **ISITMGD FAILED - RC=*rc* RS=*rs* FOR FILE *file* AND DATA SET *dsn***

section

An ISITMGD macro was issued against a program library, but ISITMGD issued a non-zero return code.

In the message text:

rc
the decimal return code issued by ISITMGD.

rs
the decimal reason code issued by ISITMGD.

file
the name of the file being processed by ISITMGD.

dsn
the name of the data set being processed by ISITMGD.

section:
The named data set is not processed, and processing continues with the next relevant data set.

section:
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 the appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSITAGP

HSIT025I ***pgmcnt* PROGRAMS FOUND TO TAG FROM DATA SET *dsn***

section

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:

pgmcnt
the number of programs processed.

dsn
the data set name containing the processed programs.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSITAGP

HSIT026I **PROCESSING COMPLETE - RC=*rc* AND *pgmcnt* PROGRAMS TAGGED IN TOTAL**

section

The product tagging utility program HSITAGP has completed processing. This message reports the return code issued by HSITAGP, and the number of programs from which data has been collected during this run.

In the message text:

rc
the return code issued by the HSITAGP upon termination.

pgmcnt
the number of programs processed in this run of HSITAGP.

section:
Processing is completed with the displayed return code.

section:
None required.

section:
None.

section:
HSITAGP

HSIT028W **UNABLE TO ACQUIRE ANY PRODUCT MAINTENANCE LEVEL DATE**

section:
After having processed all of the relevant programs, HSITAGP was unable to acquire any date stamp for use as a maintenance level indicator.

section:
Blanks are placed in the maintenance level field and processing continues.

section:
None required.

section:
None.

section:
HSITAGP

HSIT029S ***stattyp*** **STATEMENT VALUE LENGTH EXCEEDS THE ALLOWED MAXIMUM OF** ***max*** **BYTES**

section
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:

stattyp
the type of input statement being processed.

max
number of bytes.

section:
Processing terminates.

section:
Correct the input and rerun the job.

section:
None.

section:
HSITAGP

HSIT030S **INVALID TEXT CHARACTER X''*char*'' FOUND IN** ***stattyp*** **STATEMENT**

section
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 can 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:

char
the hexadecimal value of the invalid text code point.

stattyp
the type of input statement being processed.

section:
Processing terminates.

section:
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.

section:
None.

section:
HSITAGP

HSIT032E **NO PROGRAMS ELIGIBLE FOR TAGGING COULD BE FOUND**

section
No programs met the selection criteria and so the request had no operation to perform.

In the message text:

section:
Processing terminates with condition code 8.

section:
If appropriate, modify the request parameters and resubmit the job.

section:
None.

section:
HSITAGP

HSIT033W **NOT ALL PROGRAM NAME INCLUSION MASKS WERE MATCHED**

section
One or more program name inclusion masks did not match any program names in the library.

In the message text:

section:
Processing continues.

section:
Correct or remove the inoperative selection filter.

section:
None.

section:
HSITAGP

HSIT035I **DUPLICATE parm VALUE IGNORED: data.**

section
A subparameter value was encountered which matched a value previously specified for the same parameter. The duplicate specification is discarded.

In the message text:

parm
name of the parameter.

data
the value found to have been repeated.

section:
Processing continues.

section:
If the value is incorrect then correct it. To prevent the message being issued remove duplicate value specifications.

section:
None.

section:
HSITAGP

HSIT999U **HSIMSG/HSITMSG FAILURE - MSGID=*msgid* RC=*rc* RS=*rs***

section
HSIMSG was called to produce a message text, but the call failed.

In the message text:

msgid
identifier of the failing message.

rc
HSIMSG return code.

rs
HSIMSG reason code.

section:
Terminates with a condition code of 20.

section:
Inform the system programmer.

section:
Ensure JOBLIB/STEPLIB contains the library where the HSITMSG message module resides. If you cannot resolve this issue, contact IBM support.

section:
HSITAGP

HSIX - Inquisitor for z/OS UNIX messages and codes

These messages are issued by the Inquisitor for z/OS and UNIX.

Return codes

Table 115. Return codes and their meaning

Return code	Description
0	No errors encountered. All requests processed successfully.
4	Warning issued.
8	Error - Incomplete data. Processing continues. OPEN or other system service error.
12	Syntax error.
16	Unrecoverable error.
20	The zip I/O subroutine returned an error indication.

Message suffix codes

Table 116. Message suffix codes and associated condition 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.

HSIX002I THE SPECIFIED DIRECTORY NAME DOES NOT START WITH A SLASH

section:
A record from file INQXROOT or file INQXOMIT was read and was found to start with a non-blank that is not a slash. It is reported in case processing errors result from the non-standard directory name.

section:
Processing continues.

section:
Correct the input if it is incorrect.

section:
None.

section:
HSIXINQ

HSIX003I PROGRAM PARAMETER "*parm*" DISCARDED

section
The program parameter contained some unrecognized data.

In the message text:

parm
parameter in error.

section:
The displayed part of the program parameter is ignored.

section:
Correct the program parameter.

section:
None.

section:
HSIXINQ

HSIX004I FUNCTION *func* COMPLETED WITH RC=*rc* AND REASON=*rs*

section
The named z/OS UNIX System Service issued a negative return value.

In the message text:

func
function name.

rc
hexadecimal return code.

rs
hexadecimal reason code.

section:
Processing continues.

section:
Determine the meaning of the return and reason codes and correct the problem if appropriate. Information relating to the failing UNIX function can be found in the *UNIX System Services Assembler Callable Services* manual. Information relating to the return code and reason code of the failing UNIX function can be found in the *UNIX System Services Messages and Codes* manual.

section:

None.

section:
HSIXINQ

HSIX006E RENAME FAILED FOR DATA SET *dsn*

section
The rename operation to add one or more extra low-level qualifiers to a data set name as specified by the LLQ program parameter setting did not succeed. The named data set is allocated to either the INQXZIP or INQXOUT file. If this message is not followed by an associated explanatory message then an IDCAMS report detailing the results of the rename attempt will have been written to SYSPRINT.

In the message text:

dsn
name of the INQXZIP or INQXOUT data set.

section:
The output data set retains its original name.

section:
Ensure that the specified LLQ string length does not exceed 44 bytes, that any symbols used are valid for this system, and that resultant data set names are not longer than 44 bytes. Examine associated messages to determine the reason for the rename failure.

section:
None.

section:
HSIXINQ

HSIX007E FUNCTION *func* FAILED, RC=*rc*, REASON=*rs*, FOR PATH *pth*

section
The named z/OS UNIX system service issued a negative return value.

In the message text:

func
function name.

rc
hexadecimal return code.

rs
hexadecimal reason code.

pth
path in error.

section:
Processing continues.

section:
Determine the meaning of the return and reason codes and correct the problem if appropriate. Information relating to the failing UNIX function can be found in the *UNIX System Services Assembler Callable Services* manual. Information relating to the return code and reason code of the failing UNIX function can be found in the *UNIX System Services Messages and Codes* manual.

section:
None.

section:
HSIXINQ

HSIX008E FUNCTION *func* WAS DENIED ACCESS TO PATH *pth*

section
The named z/OS UNIX System Service issued a return code of hexadecimal 6F which indicates that access was denied.

In the message text:

func
function name.

pth
path in error.

section:
Processing continues.

section:
Grant the user access to the parts of the UNIX file system to be scanned.

section:
None.

section:
HSIXINQ

HSIX009S NO EXECUTABLE SOFTWARE FOUND - NO DATA FOR IMPORT WAS PRODUCED

section:
All scanning operations failed to find any programs or other executable software, so no data suitable for subsequent processing was created.

section:
Terminates with a condition code of 12.

section:
Correct any selection criteria errors and rerun the job.

section:
None.

section:
HSIXINQ

HSIX010U **MISSING INQXOUT AND INQXZIP FILES**

section:
Neither an INQXOUT nor an INQXZIP file is allocated. At least one output file is required.

section:
Terminates with a condition code of 16.

section:
Specify an output file and rerun the job.

section:
None.

section:
HSIXINQ

HSIX999U **INQXMSG FAILURE - MSGID=*msgid* RC=*rc* RS=*rs***

section
HSIMSG was called to produce a message text, but the call failed.

In the message text:

msgid
identifier of the failing message.

rc
HSIMSG return code.

rs
HSIMSG reason code.

section:
Terminates with a condition code of 20.

section:
Inform the system programmer.

section:
Ensure JOBLIB/STEPLIB contains the library where the INQXMSG message module resides. If you cannot resolve this issue, contact IBM support.

section:
HSIXINQ

HSIZ - Usage Monitor messages

These messages are issued by the Usage Monitor.

Return codes

Table 117. Return codes and their meaning

Return code	Description
0	Normal termination.
16	Initialization failed.

Message suffix codes

Table 118. Message suffix codes and associated condition 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.

HSIZ001I **USAGE MONITOR INITIALIZING**

section:
The Usage Monitor has been started.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ002I ***csid*** **DETECTED UNSUPPORTED OPERATING SYSTEM**

section:
The Usage Monitor may not run on an unsupported operating system.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ003I ***csid*** **USAGE MONITOR NOT APF AUTHORIZED**

section
The Usage Monitor needs to be executed in an APF authorized environment.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
See System Programmer to correct the error.

section:
APF authorize the load libraries that the Usage monitor runs from.

section:
HSIZMON

HSIZ005I ***csid*** **USAGE MONITOR ALREADY ACTIVE**

section:
The Usage Monitor is already running. Only one concurrent copy can run in an operating system image.

In the message text:

csid
current system identifier.

section:
Processing terminates. The established Usage Monitor task continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ006I ***csid*** **USAGE MONITOR QEDIT BUFFER SET FAILED**

section:
A QEDIT issued to set up MODIFY command processing has failed.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
Notify the system programmer.

section:
Gather appropriate diagnostic materials and contact IBM support.

section:
HSIZMON

HSIZ007I ***csid*** **USAGE MONITOR MODULE** ***mod*** **FAILED - RC=***rc*

section:
A Usage Monitor subroutine has failed.

In the message text:

csid
current system identifier.

mod
failing module name.

rc
decimal return code.

section:
Processing terminates.

section:
Notify the system programmer.

section:
Gather appropriate diagnostic materials and contact IBM support.

section:
HSIZMON

HSIZ008I ***csid*** **USAGE MONITOR INITIALIZED - ASID *asid* SET IN AVT *avt***

section:
An anchor vector table (AVT) has been acquired or reacquired and has been updated for the current server address space, which has completed initialization.

In the message text:

csid
current system identifier.

asid
ASID number.

avt
AVT Address.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ009I **DATA WRITTEN TO DSN=*dsn***

section:
Usage Monitor data has been written to the named data set.

In the message text:

dsn
data set name of the created output.

section:
Processing continues.

section:
Transfer the named data set to the system where the database resides so it can be processed.

section:
None.

section:
HSIZMON

HSIZ010E ***csid*** **USAGE MONITOR - WRITER TASK ENDED - RC=*rc***

section:
A writer task has ended with a non-zero return code.

In the message text:

csid
current system identifier.

rc
return code of writer task.

section:
Processing continues.

section:
Notify the system programmer.

section:
Gather appropriate diagnostic materials and contact IBM support.

section:
HSIZMON

HSIZ011E ***csid*** **USAGE MONITOR - WRITER TASK ABENDED - *abend***

section:
A writer task has ended abnormally.

In the message text:

csid
current system identifier.

abend
abend code from writer task.

section:
Processing continues.

section:
Notify the system programmer.

section:
Local reasons for system abends should be investigated. If necessary, gather appropriate diagnostic materials and contact IBM support.

section:
HSIZMON

HSIZ012I ****DATA LOSS** UNUSABLE DSN=*dsn***

section:
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:

dsn
data set name of the created output file.

section:
Processing continues.

section:
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, 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.

section:
Investigate any writer task abends.

section:
HSIZMON

HSIZ013I ***csid*** **USAGE MONITOR - UNRECOGNIZED PROGRAM PARAMETER IGNORED**

section:
An unrecognized program parameter was specified.

In the message text:

csid
current system identifier.

section:
Processing continues.

section:
Remove or correct the program parameter.

section:
None.

section:
HSIZMON

HSIZ014I ***csid*** **USAGE MONITOR - COULD NOT OPEN FILE UMONIN**

section:
The UMONIN file could not be opened by the Usage Monitor.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
Supply or correct the UMONIN DD statement in the JCL.

section:
None.

section:
HSIZMON

HSIZ015I ***csid*** **USAGE MONITOR - COULD NOT OPEN FILE UMONMSG**

section:
The UMONMSG file could not be opened by the Usage Monitor.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
Supply or correct the UMONMSG DD statement in the JCL.

section:
None.

section:
HSIZMON

HSIZ016I ***csid*** **USAGE MONITOR TERMINATING - INVALID OR MISSING UMONIN DATA**

section:
At least one UMONIN input statement was invalid, or input required to be present in the UMONIN file was missing.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
Examine the UMONMSG output report. Correct any invalid statements. Ensure a valid data set name prefix was specified.

section:
None.

section:
HSIZMON

HSIZ017I ***csid*** **USAGE MONITOR TERMINATING - NOW WRITING CAPTURED DATA**

section:
A STOP command has been encountered. The current repository contents are written before the Usage Monitor terminates.

In the message text:

csid
current system identifier.

section:
The Usage Monitor starts a writer task and waits for its completion before terminating.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ018I ***csid*** **USAGE MONITOR HAS NOW TERMINATED**

section:
The Usage Monitor has now freed resources and is about to terminate.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ019I ***csid*** **USAGE MONITOR REPOSITORY FULL - NOW SWITCHING**

section:
The current Usage Monitor data collection repository is full.

In the message text:

csid
current system identifier.

section:
A new repository is created and used for subsequent data collection. A writer task is initiated for the full repository.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ020I ***csid*** **THE SPECIFIED NUMBER WAS TOO SMALL**

section:
The numeric value of a command subparameter was too small to be valid in the command context.

In the message text:

csid
current system identifier.

section:
The command is discarded.

section:
Correct the numeric value and reissue the command.

section:
None.

section:
HSIZMON

HSIZ021I ***csid*** **THE SPECIFIED NUMBER WAS TOO LARGE**

section:
The numeric value of a command subparameter was too large to be valid in the command context.

In the message text:

csid
current system identifier.

section:
The command is discarded.

section:
Correct the numeric value and reissue the command.

section:
None.

section:
HSIZMON

HSIZ022I ***csid*** **PASSIVE MODE SET FROM PROGRAM PARAMETER**

section:
PASSIVE was specified in the program parameter.

In the message text:

csid
current system identifier.

section:
The Usage Monitor starts in passive mode unless overridden by input from the UMONIN file.

section:
Set the Usage Monitor into collection mode to start data collection.

section:

None.

section:
HSIZMON

HSIZ023I ***csid*** **PROGRAM NAME MASK** ***mask*** **NOT ADDED - ALREADY IN TABLE**

section:
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:

csid
current system identifier.

mask
program mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ024I ***csid*** **PROGRAM NAME MASK** ***mask*** **ADDED TO TABLE**

section:
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:

csid
current system identifier.

mask
program mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ025I ***csid*** **PROGRAM NAME MASK** ***mask*** **NOT DELETED - NOT FOUND IN TABLE**

section:
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:

csid
current system identifier.

mask
program mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ026I ***csid*** **PROGRAM NAME MASK** ***mask*** **DELETED FROM TABLE**

section:
A command to delete a program name mask from a program mask table was issued, and the mask was deleted successfully.

In the message text:

csid
current system identifier.

mask
program mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ027I **ECSA APPEARS TO BE EXHAUSTED - INCREASE SIZE FOR NEXT IPL**

section:
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.

section:
Processing continues.

section:
Notify the system programmer.

section:
Add approximately 50 to 100 MB to the ECSA size in the system IPL parameters. Check the capacity of the COMMON page data set.

section:
HSIZMON

HSIZ028I ECSA AND CSA APPEAR TO BE EXHAUSTED - INCREASE ECSA NEXT IPL

section:
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.

section:
Processing continues.

section:
Notify the system programmer.

section:
Add approximately 50 to 100 MB 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.

section:
HSIZMON

HSIZ029I *csid* THERE IS CURRENTLY NO EXCLUDE TABLE

section:
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:

csid
current system identifier.

section:
Processing continues.

section:
None required. The EXC command may be used to create a table.

section:
None.

section:
HSIZMON

HSIZ030I *csid* USAGE MONITOR - NO DATA COLLECTED SO SKIPPING WRITE

section:
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:

section

csid
current system identifier.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ032I *csid cmd* COMMAND UNKNOWN

section:
A command was issued but was not recognized.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

section:
The command is ignored. Processing continues.

section:
If necessary, correct and reissue the command.

section:
None.

section:
HSIZMON

HSIZ033I *csid cmd* COMMAND PROCESSED

section:
A command was issued and has been processed successfully.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

If the system identifier is shown as **UMON-WTR** then the command was issued internally by the writer task. Look under the description of the command for more information.

section:

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ034I ***csid cmd*** **COMMAND HAS INVALID OPERAND**

section:
A command was issued but an invalid operand was encountered.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

section:
The command is ignored. Processing continues.

section:
If necessary, correct and reissue the command.

section:
None.

section:
HSIZMON

HSIZ035I ***csid cmd*** **COMMAND FAILED**

section:
A command was issued but insufficient resources were available to execute it successfully.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

section:
The command is ignored. Processing continues.

section:
Try again after more resources become available.

section:
None.

section:
HSIZMON

HSIZ036I ***csid cmd*** **COMMAND CAUSED NO CHANGE**

section:
A command was issued but the state to be set by the command was found to already exist.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ037I ***csid cmd*** **COMMAND REJECTED**

section:
A recognized command was issued at a time when the Usage Monitor is unable to process the command.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

section:
The command is ignored. Processing continues.

section:
Try again after the Usage Monitor has freed the resources.

section:
None.

section:
HSIZMON

HSIZ038I ***csid*** **CURRENT USAGE MONITOR PROGRAM EXCLUDE LIST**

section:
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:

csid
current system identifier.

section:
The data is displayed and processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ039I ***csid*** **REPOSITORY SWITCH HAS BEEN QUEUED**

section:
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" on page 283, which shows the creation timestamp of the active writer task.

In the message text:

csid
current system identifier.

section:
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.

section:
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.

section:
None.

section:
HSIZMON

HSIZ040I ***csid*** **WAITING FOR WRITER TASK ATTACHED** ***ts***

section:
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" on page 283 and shows the creation timestamp of the active writer task.

In the message text:

csid
current system identifier.

ts
Time stamp of write task.

section:
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.

section:
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.

section:
None.

section:
HSIZMON

HSIZ041I ***csid*** **CURRENT USAGE MONITOR OUTPUT DYNALLOC PARMS**

section:
A D-A command was issued to display the current output dynamic allocation parameters, which are shown after this message.

In the message text:

csid
current system identifier.

section:
The data is displayed and processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ042I **CURRENT USAGE MONITOR OUTPUT SYSTEM ID IS** ***csysplex.csid***

section:
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:

csysplex
current sysplex name.

csid
current system identifier.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ043I ***csid*** **DATA DISCARDED DUE TO QUEUE AREA STORAGE LIMIT**

section:
The Usage Monitor has detected for the first time in the life of the current collection repository that program usage event data has been discarded due to the queue area storage usage limit being reached. This limit was set with the QSZ command.

In the message text:

csid
current system identifier.

section:
Processing continues.

section:
Adjust the Usage Monitor QSZ setting as appropriate for the particular system. Recycle the Usage Monitor address space to implement the new setting. 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.

section:
None.

section:
HSIZMON

HSIZ044I ***csid*** **SWITCH-AND-WRITE TIME-OF-DAY IS SET TO** ***hh*****:*****mm***

section:
A D-T command was issued to display the switch-and-write time-of-day setting for this system.

In the message text:

csid
current system identifier.

hh
Hour of the day.

mm
minute of the hour.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ045I ***csid*** **CREATED** ***area token-addr***

section:
A memory object required to hold data was created.

In the message text:

csid
current system identifier.

area
purpose of the memory object.

token
storage token of the memory object.

addr
storage address of the memory object.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ046I ***csid*** **DELETED** ***area token-addr***

section:
A memory object which was no longer needed was deleted.

In the message text:

csid
current system identifier.

area
purpose of the memory object.

token
storage token of the memory object.

addr
storage address of the memory object.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ047I ***csid*** **USAGE MONITOR - ATTACHING WRITER SEQ-NO-** ***seqnbr***

section:
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:

csid
current system identifier.

seqnbr
sequence number of writer task this run.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ048I ***csid*** **USAGE MONITOR - IDENTIFY FAILED HEX RC=*rc***

section:
The Usage Monitor executed an IDENTIFY macro which failed.

In the message text:

csid
current system identifier.

rc
hexadecimal return code of the IDENTIFY macro.

section:
Processing terminates.

section:
Notify the system programmer.

section:
Investigate why an IDENTIFY macro would fail with that return code.

section:
HSIZMON

HSIZ049I ***csid*** **DATA SET NAME MASK NOT DEACTIVATED, NOT FOUND IN LIST**

section:
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:

csid
current system identifier.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ050I ***csid*** **DATA SET NAME MASK *mask* LIST *list***

section:
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:

csid
current system identifier.

mask
INCLUDE or EXCLUDE.

list
START or END.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ053I ***csid*** **MONITORING UNIX PROGRAMS?** ***ans***

section:
Either a UNX 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:

section

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ054I ***csid*** **MONITORING LINK PACK AREA PROGRAMS?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ056I ***csid*** **PREFER VOLUME SYMBOL OVER SERIAL?** ***ans***

section:
Either a SYM command was issued to change the volume symbol status or a D-S command was issued.

- When the answer is YES, a matching system static symbol that evaluates to the volume serial is collected instead of the volume serial, if such a symbol exists; otherwise the actual volume serial is collected. A YES setting might improve data matching when system software platform volume switches take place.
- When the answer is NO, the captured volume serial number is always output.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ058I ***csid*** **FILE UMONIN IS NOT ALLOCATED - CANNOT PERFORM REFRESH**

section:
A REF command was issued to refresh settings from commands in the UMONIN file, but the UMONIN file had been freed, and was no longer allocated to the Usage Monitor.

In the message text:

csid
current system identifier.

section:
The refresh operation is suppressed and processing continues.

section:
Ensure FREE=CLOSE is not specified in the UMONIN JCL DD statement. Recycle the Usage Monitor to refresh the settings if necessary.

section:
None.

section:
HSIZMON

HSIZ059I ***csid*** **REFRESH PERFORMED WITH NO ERRORS**

section:
A REF command was issued to refresh settings from commands in the UMONIN file. All commands in the UMONIN file were completed successfully.

In the message text:

section

csid
current system identifier.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ060I ***csid*** **REFRESH PERFORMED BUT ERROR(S) FOUND**

section:

A REF command was issued to refresh settings from commands in the UMONIN file. At least one command in the UMONIN file resulted in an error.

In the message text:

csid
current system identifier.

section:
Processing terminates.

section:
Examine the output in the UMONMSG file to determine the problem(s).

section:
None.

section:
HSIZMON

HSIZ063I ***csid*** **COLLECTING "UNKNOWN" EVENTS?** ***ans***

section:
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:

section

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ064I ***csid*** **WILL WRITER TASK COMPRESS THE DATA?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ065I ***csid*** **WILL WRITER TASK CORRECT LINKLIST DSN?** ***ans***

section:
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:

section

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ066I ***csid nbr*** **IDLE ELEMENT(S) "LOST" DUE TO ZERO POINTER**

section:
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:

csid
current system identifier.

nbr
the number of elements being reported.

section:
Termination continues.

section:
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.

section:
None.

section:
HSIZMON

HSIZ068I ***csid*** **COLLECTING JOB ACCOUNTS NOW?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ069I ***csid*** **COLLECTING JOB ACCOUNTS LATER?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ070I ***csid*** **COLLECTING REGISTERED PRODUCT DATA NOW?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ071I ***csid*** **COLLECTING REGISTERED PRODUCT DATA LATER?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:

None.

section:
None.

section:
HSIZMON

HSIZ072I ***csid*** **COLLECTING DYNAMIC CAPACITY DATA NOW?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ073I ***csid*** **COLLECTING DYNAMIC CAPACITY DATA LATER?** ***ans***

section:
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:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ074I ***csid*** **OUTPUT NAMES OF COLLECTED USERS?** ***ans***

section:
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. Even if the answer is YES, no user names will be output if no user information was collected.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ075I ***csid*** **COLLECTING USER INFORMATION NOW?** ***ans***

section:
A D-S command was issued. When the answer is YES, the identifier and name of each program user is currently being collected by the Usage Monitor. When the answer is NO, these user details are not being currently collected.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ076I ***csid*** **COLLECTING USER INFORMATION LATER?** ***ans***

section:

Either a UID command was issued or a D-S command was issued. When the answer is YES, the identifier and name of each program user will be collected after the next Usage Monitor collection repository switch. When the answer is NO, these userid details will not be collected after the next switch.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ077I ***csid* COLLECTING JOB NAMES NOW? *ans***

section:
A D-S command was issued. When the answer is YES, the names of jobs using programs are currently being collected by the Usage Monitor. When the answer is NO, only generic address space type data such as JOB, STC, and TSU is currently being collected instead of individual job names.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ078I ***csid* COLLECTING JOB NAMES LATER? *ans***

section:
Either a JNM command was issued or a D-S command was issued. When the answer is YES, the names of jobs using programs will be collected after the next Usage Monitor collection repository switch. When the answer is NO, only generic address space type data such as JOB, STC, and TSU will be collected after the next switch instead of individual job names.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZMON

HSIZ079I ***csid* ALLOWING CICS EVENT COLLECTION? *ans***

section:
A D-S command was issued. When the answer is YES, this message indicates that CICS event collection from suitably customized CICS regions is enabled. When the answer is NO, CICS generated program usage information will not be collected.

In the message text:

section

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ080I ***csid dsn - mct* MATCHES**

section:
Displays the dataset name mask for a D-D command.

In the message text:

csid
current system identifier.

dsn
data set name.

mct
match count.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ081I ***csid*** **MONITORING PREVIOUSLY RUNNING PROGRAMS?** ***ans***

section:
A D-S command was issued. When the answer is YES, this message indicates usage data for programs resident in the regions of jobs which are older than the current Usage Monitor repository and which have SMF interval recording active will be collected. When the answer is NO, there will be no usage data collected for programs that were running before the current repository was created and do not terminate before the repository collection period ends.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ082I ***csid*** **RETAINING ALL BATCH JOB IDENTIFIERS?** ***ans***

section:
Either a JID or a D-S command was issued. When the answer is YES, the message indicates that data for batch jobs with different JES job identifiers will not be aggregated together but will be reported as usage by separate jobs. When the answer is NO, data for usage of programs will be aggregated only by userid and job name, and only the most recent job identifier will be retained. Aggregation corresponding to the NO answer is always used for started tasks and TSO sessions.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ083I ***csid*** **ADJUSTING FOR HYPERVISOR TIME OFFSET?** ***ans***

section:
Either a HOF or a D-S command was issued. When the answer is YES, the message indicates that timestamps in collected data will be adjusted by subtracting the hypervisor time offset from the z/OS date and time. When the answer is NO, the unaltered local z/OS date and time will be used in the collected data.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ084I ***csid*** **REPORTING TCPIP HOST DETAILS?** ***ans***

section

Either an IPH or a D-S command was issued. When the answer is YES the message indicates that the TCPIP host details (name and address) will be reported in the usage data file. When the answer is NO then the Usage Monitor will not report any TCPIP host information.

In the message text:

csid
current system identifier.

ans

YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ085I ***csid cmd*** **NO LONGER SUPPORTED**

section:
A command was issued that is no longer a functional Usage Monitor command.

In the message text:

csid
current system identifier.

cmd
name of the issued command.

section:
The command is ignored. Processing continues.

section:
Do not issue this command.

section:
None.

section:
HSIZMON

HSIZ086I ***csid*** **COLLECTING APF LIBRARY DETAILS?** ***ans***

section:
Either an APF or a D-S command was issued. When the answer is YES the message indicates that the volumes and data set names of APF authorized libraries will be collected and written in the output data. This information is not collected when the answer is NO.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ087I ***csid*** **ACTIVE LONG-RUNNING PROGRAM NAME LIST**

section:
A D-L command was issued to display the long-running program name list. The active entries are shown after this message.

In the message text:

csid
current system identifier.

section:
The data is displayed and processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ088I ***csid*** **COLLECTING USAGE FROM TEMPORARY DATA SETS?** ***ans***

section:
Either a TMP or a D-S command was issued. When the answer is YES the message indicates that usage from programs fetched from temporary data sets will be collected and written in the output data. Usage data about these programs will be discarded when the answer is NO.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ089I ***csid*** **ALLOWING IMS USAGE COLLECTION?** ***ans***

section:
A D-S command was issued. When the answer is YES this message indicates that usage of IMS-managed programs will be collected. Usage of IMS-managed programs will not be collected when the answer is NO.

In the message text:

csid
current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ090I ***csid*** **ALL** ***type*** **ENTRIES HAVE BEEN DEACTIVATED**

section:
A DEL, IDD or XDD command was issued with *ALL* being the only specified mask. As a result, all of the relevant type of filtering mask entries were deactivated. The filtering made inactive by the DEL, IDD or XDD command is of the type set by EXC, IDS or XDS commands, respectively.

In the message text:

csid
current system identifier.

type
EXC, IDS or XDS.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ091I ***csid*** **DEFAULT** ***type*** **EXCLUSION ENTRIES ARE NOW ACTIVE**

section:
An EXC or XDS command was issued with *DFLT* being the only specified mask. As a result, all of the relevant type of default exclusion mask entries are now active. When the reported type is PROGRAM, program name filtering has been reset to the shipped default. When the reported type is DATA SET, all the default data set exclusion entries have been made active without altering user specified data set name filtering.

In the message text:

csid
current system identifier.

type
PROGRAM or DATA SET.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ092I ***csid*** **COLLECTING LINKLIST LIBRARY DETAILS?** ***aus***

section:
Either a LNK or a D-S command was issued. When the answer is YES the message indicates that the volumes and data set names of linklist libraries will be collected and written in the output data. This information is not collected when the answer is NO.

In the message text:

csid

current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ094I ***csid*** **COLLECTING NAMES MADE BY IDENTIFY?** ***ans***

section:
A D-S command was issued. When the answer is YES the message indicates IDY(Y) has been set, so entry point names created by the IDENTIFY macro will be reported in usage data even though collected names may not match names in software inventory. When the answer is NO the message indicates that usage for entry point names created by IDENTIFY will either be ignored or reported as usage of real programs.

In the message text:

csid

current system identifier.

ans
YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ101I ***csid*** **CURRENT TRACE JOB NAME MASK SELECTION LIST**

section:
A TRCD-J command was issued to display the trace job name mask selection list. The active entries are shown after this message.

In the message text:

csid

current system identifier.

section:
The data is displayed and processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ102I ***csid*** **CURRENT TRACE PROGRAM NAME MASK SELECTION LIST:**

section:
A TRCD-P command was issued to display the trace program name mask selection list. The active entries are shown after this message.

In the message text:

csid

current system identifier.

section:
The data is displayed and processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ103I ***csid*** **CURRENT USAGE MONITOR TRACE OUTPUT DYNALLOC PARMS:**

section

A TRCD-A command was issued to display the current trace output dynamic allocation parameters, which are shown after this message.

In the message text:

csid

current system identifier.

section:
The data is displayed and processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ104I ***csid*** **JOB NAME MASK** ***mask*** **ADDED TO TABLE**

section

A command to add a job name mask to a job mask table was issued, and the mask was added successfully.

In the message text:

csid

current system identifier.

mask

job name mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ105I ***csid*** **USAGE MONITOR - TRACE TASK ENDED - CC=** ***cc***

section

The Usage Monitor trace subtask has ended with the displayed completion code.

In the message text:

csid

current system identifier.

cc

completion code of the trace subtask.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ106I ***csid* USAGE MONITOR - TRACE TASK ABENDED - *code***

section

The Usage Monitor trace subtask has ended abnormally with the displayed abend code.

In the message text:

csid

current system identifier.

code

abend code of the trace subtask.

section:
Processing continues.

section:
Notify the system programmer.

section:
Local reasons for the abend should be investigated. If necessary, gather appropriate diagnostic materials and contact HCL support.

section:
HSIZMON

HSIZ107I ***csid* USAGE MONITOR - ATTACHING TRACE SEQ-NO- *seqnbr***

section

A trace task is being attached to capture event records. The trace task sequence number is also reported. The first trace task to run after the Usage Monitor starts has a sequence number of 1.

In the message text:

csid

current system identifier.

seqnbr

sequence number of trace task.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ108I ***csid* TRACE CURRENTLY ACTIVE? *ans***

section

A D-S command was issued. When the answer is YES a trace output data set is allocated and is accruing data based on trace selection filters and program usage activity. When the answer is NO, no program usage events are being separately logged as trace data.

In the message text:

csid

current system identifier.

ans

YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ109I ***csid* TRACING EXCLUDED EVENTS? *ans***

section

Either a TRCEXC or a D-S command was issued. When the answer is YES the message indicates that events excluded from the normal data collection are eligible to be traced subject to the active trace selection filters. When the answer is NO events excluded from normal data collection are also excluded from trace data collection even if they otherwise match trace selection criteria.

In the message text:

csid

current system identifier.

ans

YES or NO.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ110I ***csid*** **JOB NAME MASK** ***mask*** **NOT DELETED - NOT FOUND IN TABLE**

section:
A command to delete a job name mask from a job mask table was issued, but the mask was not present in the table.

In the message text:

section

csid

current system identifier.

mask

job name mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ111I ***csid*** **JOB NAME MASK** ***mask*** **DELETED FROM TABLE**

section:
A command to delete a job name mask from a job mask table was issued, and the mask was deleted successfully.

In the message text:

csid

current system identifier.

mask

job name mask specified in command.

section:
Processing continues.

section:
None required.

section:
None.

section:
HSIZMON

HSIZ201I **DYNALLOC FAILURE RC=** ***rc*** **ERROR=** ***s99error*** **INFO=** ***s99info*** **DSN=** ***dsn***

section:
The writer task could not dynamically allocate a new output data set.

In the message text:

rc
DYNALLOC return code.

s99error
dynamic allocation reason code (DARC).

s99info
dynamic allocation information code.

dsn
name of the data set being allocated.

section:
Processing of the repository is terminated, and the data lost.

section:
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 the appendix of the ISPF Tutorial.

section:
None.

section:
HSIZ0203

HSIZ202I **USAGE MONITOR - COMPRESSION SUBROUTINE ERROR**

section:
While processing repository data the compression subroutine encountered an error. The error message from the compression subroutine immediately follows this message.

section:
Processing of the repository is terminated, and the data lost.

section:
Correct the error described in the message from the compression subroutine. If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
None.

section:
HSIZ0203

HSIZ206I ***errmsg***

section:
The HSISHRNK compression routine issued an error message which is displayed.

In the message text:

errmsg
error message from HSISHRNK.

section:
The message is preceded by message "HSIZ202I" on page 296.

section:
Examine the message for further information.

section:
None.

section:
HSIZ0203

HSIZ301I **DESERV FUNC=EXIT RC=*rc* REASON=*rs***

section:
DESERV FUNC=EXIT issued a non-zero return code.

In the message text:

rc
return code from DESERV.

rs
reason code from DESERV.

section:
The DESERV exit is not installed.

section:
Notify the system programmer.

section:
Research the DESERV feedback to determine why the exit could not be installed.

section:
HSIZ0203

HSIZ302I **CSVDYNEX ADD (*excd*) RC=*rc* REASON=*rs***

section:
CSVDYNEX ADD issued a non-zero return code. An exit could not be dynamically defined for the named exit point.

In the message text:

excd
dynamic exit point name.

rc
return code from CSVDYNEX.

rs
reason code from CSVDYNEX.

section:
The SMF exit is not installed.

section:
Notify the system programmer.

section:
Research the CSVDYNEX feedback to determine why the exit could not be installed. Ensure that IEFU84 is an active SMF exit for the system or subsystem. If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
HSIZ0203

HSIZ303I **ATTRIBUTE MISMATCH - *mod* NOT INSTALLED**

section:
The examined SVC table entry did not have the expected attributes.

In the message text:

mod
module name.

section:
The SVC intercept is not installed.

section:
Notify the system programmer.

section:
Gather appropriate diagnostic materials and contact IBM support.

section:
HSIZ0203

HSIZ306I **BAD *statnm* ENTRY PGM=*pgm* JOB=*jbn* USER=*user* ID=*id* DATE=*date* REJECTED**

section:
An invalid work element has been detected and some of its contents are displayed.

In the message text:

statnm
status name.

pgm
program name.

jbn
job name.

user
user name.

id
id name.

date
date.

section:
Attempted to dump some data to HSIZSNAP if the file is allocated, and will then try to free the work element without processing its contents.

section:
Notify the system programmer.

section:
The problem is indicative of a storage overlay. Gather appropriate diagnostic materials and contact IBM support.

section:
HSIZ0203

HSIZ307I ***csid*** **SCAN OF** ***volume*** **VTOC ABENDED -S** ***cde***

section:
A scan of the named volume abended with the reported abend code. The scan was performed to find the data set name of a program library that is in use.

In the message text:

csid
current system identifier.

volume
volume where VTOC resides.

cde
abend code from VTOC scan.

section:
Processing continues.

section:
Notify the system programmer.

section:
Check the named volume for error conditions. If necessary, gather appropriate diagnostic materials and contact IBM support.

section:
HSIZMON

HSIZ310I **MODULE** ***mod*** **INSTALLED AT ADDRESS** ***loadpt*** **SIZE** ***size***

section:
The Usage Monitor has dynamically loaded a module into common storage and will now register it in DLPA.

In the message text:

mod
module name.

loadpt
module load point.

size
module size.

section:
Processing continues.

section:
None.

section:
None.

section:
HSIZ0203

HSIZ311I **CSVDYLPA RC=***rc* **RS=***rs* **FOR** ***mod***

section:
The Usage Monitor attempted to register a newly installed module in DLPA, but CSVDYLPA issued a non-zero return code.

In the message text:

rc
decimal return code issued by CSVDYLPA.

rs
hexadecimal reason code issued by CSVDYLPA.

mod
name of the module being registered.

section:
Processing continues.

section:
Notify the system programmer.

section:
Investigate why the named module could not be registered in the current DLPA configuration.

section:
HSIZ0203

HSIZ999U **HSIMSG/HSIZMSG FAILURE - MSGID=***msgid* **RC=***rc* **RS=***rs*

section:
HSIMSG was called to produce a message text, but the call failed.

In the message text:

msgid
identifier of the failing message.

rc
HSIMSG return code.

rs
HSIMSG reason code.

section:
Terminates with a condition code of 20.

section:
Inform the system programmer.

section:
Ensure JOBLIB/STEPLIB contains the library where the HSIZMSG message module resides. If you cannot resolve this issue, gather appropriate diagnostic materials and contact IBM support.

section:
HSIZMON

Appendix A: Description of output message content

Messages generated during batch processing are explained in detail here.

Inquisitor

For each request based on scanning one or more VTOCs, the z/OS Inquisitor produces a VTOC Statistics Report unless the NOVSR keyword appears in the program parameter string.

The following table describes each of the data item columns shown in the report.

Table 119. Volume Statistics Report column headings

Item	Description
VOLUME	Volume serial number.
MAP-TO	Logical volume mapped to if mapped.
DVNO	Device number.
STOGROUP	SMS storage group name.
TCBADR-ID	LPAR record.
START-&-STOP-TIME	Start and stop times of volume scan by subtask.
WT-DELAY	Main task idle time waiting for this volume scan to complete.
ERROR-CD	Subtask processing error indicator.
VOL-CYL	Number of cylinders provided by the volume.
VTOCTRK	VTOC size in tracks.
IOERR	VTOC I/O error count.
DSCB	VTOC size in DSCBs.
VOL-DS	Number of data sets on the volume.
SCAN-DS	Number of data sets on the volume selected for scanning.
SCAN-PGMS	Number of scanned programs on the volume.

ZCAT

When the VFY=Y setting is active, the ZCAT program reads the main output data set (which is allocated to the ZCATOUT DD) to ensure that it can be processed. This output verification phase was added to assist with tracking file corruptions as files are transferred from systems where the data is collected to systems where the data is processed.

The ZCATOUT Verification Report produces statistics about the contents of the ZCATOUT file and issues a message regarding the success or failure of the verification process.

The messages have the following format:

count **record read of type** *type*

where *count* is the record count and *type* is the first two characters of the record. After the subtotal of each record type is reported, the total number of records in the file is displayed.

An example of the report contents is:

```
       8 records read of type UM
     151 records read of type *C
    2531 records read of type *D
      16 records read of type *E
     287 records read of type *F
     216 records read of type *L
       8 records read of type *P
     382 records read of type *R
    3599 records read from ZCATOUT file

** ZCATOUT file verify result: SUCCESS
```

If an error such as a record count mismatch is discovered, then other relevant messages may be issued, followed by this message:

```
** ZCATOUT file verify result: FAILURE
```

The step completion code is set to 16 to indicate verification failure.

Table 120. Current types of Usage Monitor records (in sequence written by Usage Monitor)

Record type	Description
UM	Header record.
*E	Header Extension record.
*F	Filter description record.
*P	Registered Product usage record.
*R	Registered Product state record.
*C	Capacity record.
*L	LPAR record.
*A	APF authorized library record.
*B	Linklist library record.
*U	User name record.
*D	Detail program usage record.

IQIMPORT

When each library is matched, several statistical messages are generated that tell you what has transpired during the matching process. See the following example:

```
modules(processed:8,ignored:0),matched(product:0,UNKNOWN:0,none:8),loaded(identified:
0,unidentified:8)
```

Item	Description
Modules	**processed** - Total number of modules processed by IQIMPORT in the library. **ignored** - Total number of modules ignored.

Item	Description
Matched	**product** - How many modules are matched to a product. **UNKNOWN** - If there are modules known in the GKB that are not matched to a product then this is the number of modules. If 0, then none of the modules are defined in the GKB. **none** - Number of modules in the library known in the GKB. If 0, then all modules are known in the GKB.
Loaded	**Identified** - This figure signifies how many modules are associated to a product. **unidentified** – This indicates how many modules are not identified in the library.

As each library is matched the log indicates each scorecard for a given GKB or LKB entry and the values associated with each match. See the following example:

```
 Initial scores

==============

 G , ProdVerID = 2123G, Num LMODs Matched = 286/292, Version Ident = 0%, Product Ident
 = 97%, score = 79%
```

Item	Description
GKB entry	**G** - Indicates scorecard is taken from the GKB. **L** – Indicates entry comes from the LKB and is user defined.
ProdVerID	Each entry in the GKB has a unique id for each version of a product. This data is used by support to find out what happened during the matching process and why some entries were not taken.
Num LMODs Matched	Indicates how many load modules are matched to the data for the given PRODVER entry. In the above example, 286 modules out of 292 in the GKB are deemed to have been identified in the library out of 292 in total in the GKB.
Version Ident	Indicates the percentage of a match to a specific version. The higher the percentage the better the match.
Product Ident	In the above example, the figure indicates the product is identified but the version is not.
score	Overall score for the scorecard entry. This score is used by the match engine after all scorecards are matched to determine what product and version the modules in the library belong to.

At the end of IQIMPORT processing a summary report is created.

```
=====================================================
IQ Import statistics for SID:OMU3
=====================================================
New Libraries                                      42
Library records - imported                         42
Library records - ignored                          0
Module records - imported                          5487
Module records - ignored                           0
Modules matched (product)                          5412
Modules matched (UNKNOWN) - insufficient score     8
Modules matched (none)    - not found in KB        67
Modules migrated(identified)                       5420
Modules migrated(unidentified)                     67
CSECTs migrated                                    319
Channel Path records                               10
Control Unit records
```

Item	Description
New Libraries	Number of new libraries processed during IQIMPORT.
Library records	**imported** – The number of libraries imported into the Repository. **ignored** – The number of libraries not imported into the Repository. These could be distribution libraries that are filtered out by the import process.
Module records	**imported** – The number of modules imported into the Repository. **ignored** – The number of modules not imported into the Repository. This could be because they are in a library that is filtered out.
Modules matched	**product** – The number of modules successfully matched to a product. **UNKNOWN** – insufficient score – The number of modules known, but not enough to register as a known version. **none** – The number of modules not found in the GKB.
Modules migrated	**Identified** – The number of modules identified to a product that are migrated to the Repository. **unidentified** – The number of module unidentified that are migrated to the Repository.
CSECTs migrated	Number of CSECTs migrated to the Repository. For CSECTs to be migrated, the IQ data must be scanned with parameter SCANPGM and keyword FULLIDR otherwise no CSECT data is collected.
Channel Path records	Number of channel paths found.
Control unit records	Number of Control units found. These records are collected using the SCANDEV command in the Inquisitor.

USAGE IMPORT

When usage data is imported into a Repository, all the data imported is displayed. Multiple usage data can be imported at the same time from the same system or multiple systems. As each data is imported, a key is generated for each period.

```
Unique ID key from input file: UIMP_IZP5_0320210808000000_0012
Processed:UIMP_IZP5_0320210808000000_0012 Elapsed time: 0 Hrs, 00 Mins, 01 Secs
```

The above example indicates that the data item is the 12th item being imported from the data file. This data is probably coming from a file that has been processed by ZCAT. The elapsed time shows you how long it takes to process the file.

At the end of Usage Import processing a summary report is created. See the below example.

```
Input Record Statistics
Number of header records read                    =    8
Number of user records read                      =  123
Number of detail records read                    = 6015
Number of product usage records read             =    0
Number of hardware capacity records read         =    8
Number of product registration records read      =  445
Number of registered product usage records read  =   32
Number of machine resource records read          =   24
                                                   ----
Total number of input records read               = 6655

Earliest period in this usage file:   Dec-2018
Latest period in this usage file:     Jun-2019
```

Usage Import run completed.

Item	Description
Number of header records read	Figure indicates number of LPARs processed during Usage Import.
Number of user records read	Figure indicates total number of individual user records collected. For example, TSO ID and STC ID.
Number of detail records read	Total number of usage records processed and imported into Repository.
Number of hardware capacity records read	Figure indicates number of LPARS capacity records read. Usually one per LPAR.
Number of product registration records read	Number of records read from Usage file containing registered products.
Number of registered product usage records read	Number of records read from Usage file indicating which products have been used.
Number of machine resource records read	Resource records read from the Usage Data showing what other resources are available on the machine.

Notices

PDF dated August 29th, 2025.

This information was developed for products and services offered in the US.

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, MD-NC119
Armonk, NY 10504-1785
US

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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non- IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non- IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non- IBM products. Questions on the capabilities of non- IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM , for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM , therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Portions of this code are derived from IBM Corp. and/or HCL Ltd. sample programs.

© Copyright IBM Corp. 2013, 2018. © Copyright HCL Technologies Limited 2018, 2024.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at https://www.ibm.com/legal/us/en/copytrade.shtml.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM .

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM .

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM , the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

A

Activating the Automation Server 115
aggregating usage and discovery data
    TPARAM parameters 114
Aggregator
    aggregating usage and discovery data 114
Analyzer
    BASIC security 131
    inactivity timeout 130
    JCLLIB and PARMLIB members 128
    online login 136
    overview 3
    prerequisites 128
    reporting 127
    running in batch mode 140
    security 131
    SSL Certificates 134
    SYSTEM and SYSTEM-TLS security 132
Analyzer address space
    controlling 140
Analyzer BASIC security 131
Analyzer communication port 130
Analyzer inactivity timeout 130
Analyzer JCLLIB and PARMLIB members 128
Analyzer online mode navigation 136
Analyzer prerequisites 128
Analyzer security 131
Analyzer SYSTEM and SYSTEM-TLS security 132
APF 13
ASCDS 122
Automation Server
    control data set 116
    control data set maintenance 122
    excluding data sets 122
    required files 117
    running 116
    starting 122
    stopping 122
Automation Server action requests 117
Automation Server control data set maintenance 122
Automation Server overview 115
Automation Server Utility
    control statements 122
    examples 122
    files used by 122
    report details 122
    request type 122
    unconditionally scheduled actions 122
Automation Server Utility program 122

C

CAP - Set hardware capacity collection status 82
CICS
    monitoring usage 108
collecting SMP/E maintenance information 51
commands
    reading syntax diagrams 4
configuring a test repository database 23
CPU time
    recording 79
creating a production Repository database 25
creating a test repository database in Db2 23
creating a test repository database in SQLIte 24
customizing a CICS region to provide usage data 109

D

D-A - Display output allocation parameters 83
D-C - Display the counters and statistics 83
D-D - Display the data set name inclusion and exclusion lists 83
D-I - Display the system identifier 83
D-S - Display the status settings 84
D-T - Display the automatic switch-and-write time setting 84
D-X - Display the active exclude list 84
data sets, scanning generation 65
database changes since the product was released
    verifying 168–170
databases
    Inquisitor requests 63
Db2 authorization 13
DCB - Set output DCB attributes 85
DEL - Deleting program mask entries 85
deployment
    Inquisitor data 10
    Repository 10
    Usage event 10
designing request control statements 117
DFHSM Integration 65
DSN
    Usage Monitor command 86
DUR
    Usage Monitor command 86

E

examples, Inquisitor for z/OS 61
EXC - Adding program mask exclusion entries 87
exclusion masks
    used to reduce data 75

F

files used by Usage Monitor 74
filtering by calling program name 77
filtering by data set name 77
filtering by program name 75
filtering by UNIX pathname 78
filtering program names 74
fragments, syntax diagrams 4

G

general commands, Usage Monitor 80
generation data sets, scanning 65

H

HSIC, Operation messages 222
HSII REXX utility messages 235
HSIP, Inquisitor messages 243
HSISIVPD
    job in the JCLLIB that verifies database changes since the product was released 168–170
HSIT - Product Tagging messages 265
HSIX, Inquisitor messages 272
HSIZ, Usage Monitor messages 275

I

IDD - Deleting data set name inclusion entries 88
IDS - Adding data set name inclusion entries 88
Import Usage
    TPARAM parameters 113
importing Inquisitor data 110
importing Subcapacity reporting data 159
importing usage data 112
initial space allocation
    performance and tuning 208
Inquisitor
    overview 2
    scanning generation data sets 65
    scanning migrated libraries 64
Inquisitor data
    description 10
    import filters and matching for import 111
    importing 110
Inquisitor for z/OS
    collecting SMP/E maintenance information 51
    examples 61
    TPARAM parameters 111
Inquisitor Import
    overview 2
Inquisitor Import, running 49, 110
Inquisitor messages, HSIC 272
Inquisitor messages, HSIP 243
Inquisitor program
    SYSIN commands 54
Inquisitor requests, designing 63
installing
    REST API 142
installing and customizing
    checklist 15
    configuring test repository database 23
    Db2 database prerequisites 19
    local environment settings 19
    target libraries 18
installing REST API 142
installing target libraries 18
integration with DFHSM 65

J

JAC - Set job account collection status 90
JNM - Control the collection of job names 90

K

keywords, syntax diagrams 4

L

LE compile unit information 64
libraries, scanning migrated 64
LMOD
    reporting ownership from the CSI 162
LNK - Set linklist library collection status 91
LPA - Set link pack area program monitoring status 92

M

messages, HSIC 222
messages, HSII 235
messages, HSIP 243
messages, HSIT 265
messages, HSIX 272
messages, HSIZ 275
migrating from 2.2 to 9.1 34
migrating from 8.2 to 9.1 (Db2 database)migrating from 2.2 to 9.1 (Db2 database) 41
migrating from 8.3 to 8.3.1 (SQLite database) 38
migrating from 8.3.1 to 9.1 (SQLite database) 45
migrating from 8.3/ 8.3.1 to 9.1 (Db2 database) 35
migrating from 8.3/8.3.1 to 9.1 34
migrating to 9.1 34
migration
    8.2 to 9.1 (Db2 database) 41
    8.3 to 8.3.1 (SQLite) 38
    8.3/ 8.3.1 to 9.1 (Db2 database) 35
    post migration errors 47
migration)
    8.3.1 to 9.1 (SQLite) 45
    migrating from 2.2 to 9.1 (SQLite database) 45
    migration
        2.2 to 9.1 (SQLite) 45
monitoring usage in CICS regions 108

N

navigation, Analyzer online mode 136
Non-DB application
    Inquisitor requests 63

O

online mode navigation, Analyzer 136
Operation messages, HSIC 222
output message content description 299

P

performance and tuning 208, 209
post migration errors 47
preparing local environment settings
    installing and customizing 19
prerequisites

prerequisites *(continued)*
REST API 141
PRI - Set the data set space primary allocation 94
process flow 3
Product Tagging, HSIT 265
production Repository database
creating 25
program names, filtering 74
PRS - Set registered software activity data collection status 94

R

RACF customization 13
recording CPU time 79
REF - Refresh Usage Monitor settings 95
repeatable items, syntax diagrams 4
reporting
using Analyzer 127
reporting LMOD ownership from the CSI 162
Repository
description 10
overview 2
Repository table layouts 174
REST API
installing 142
prerequisites 141
REXX utility messages, HSII 235
running Analyzer in batch mode 140
running Usage Import 113

S

scanning migrated libraries 64
SEC - Set the data set space secondary allocation 95
security, z/OS UNIX 13
SID - Set the Usage Monitor system identifier 96
SSL Certificates 134
Subcapacity reporting
importing with SCRT Import utility 159
syntax diagrams, how to read 4
system platform
Inquisitor requests 63

T

table layouts, Repository 174
Trace data flow 101
trace data set contents 101
Trace Facility
data collected 101
TRCD-A - Display data set allocation parameters 103
TRCD-J -Display trace selection program name masks 103
TRCD-J- Display trace selection job name masks 103
TRCDCL - Set output data set SMS data class 103
TRCDML - Set output data set SMS management class 105
TRCDSN - Set output data set name 103
TRCEXC - Control tracing of excluded events 104
TRCJOB - Specify job name trace selection filter 104
TRCJOD - Delete job name trace selection filter 105
TRCOFF - Terminate tracing 105
TRCON - Initiate tracing 105
TRCPGD - Delete job name trace selection filter 106
TRCPGM - Specify program name trace selection filter 106
TRCPRI - Set output data set primary space 106
TRCSCL - Set output data set SMS storage class 106
TRCSEC - Set output data set secondary space 107
TRCSTA - Set output data set initial disposition 107
TRCUNIT – Set output data set device allocation unit 107
TRCUNX -Control tracing of UNIX syscalls 107
TRCVOL – Set output data set volume 108
troubleshooting
identifying problems 213
systematic problem solving 213
TSO
Inquisitor requests 63

U

UNIX security, z/OS 13
usage
monitoring in CICS regions 108
usage and discovery data
aggregating 114
usage data
customizing a CICS region 109
usage event
description 10
Usage Import
overview 2
running 113
Usage import job 112
Usage Monitor
data collected by Trace Facility 101
filtering by calling program name 77
filtering by data set name 77
filtering by program name 75
filtering by UNIX pathname 78
overview 2
recording CPU time 79
refresh processing 80
running 79
trace commands 102
Trace data flow 101
trace data set contents 101
Trace Facility 101
Usage Monitor command
CAP - Set hardware capacity collection status 82
CIC – Allow or disable program usage data from CICS regions 82
D-A - Display output allocation parameters 83
D-C - Display the counters and statistics 83
D-D - Display the data set name inclusion and exclusion lists 83
D-I - Display the system identifier 83
D-S - Display the status settings 84
D-T - Display the automatic switch-and-write time setting 84
D-X - Display the active exclude list 84
DCB - Set output DCB attributes 85
DEL - Deleting program mask entries 85
DSN - Setting the data set name prefix 86
DUR - Set execution duration 86
EXC - Adding program mask exclusion entries 87
IDD - Deleting data set name inclusion entries 88
IDS - Adding data set name inclusion entries 88
JAC - Set job account collection status 90

Usage Monitor command *(continued)*
JID - Control the preservation of batch job identifiers 90
JIPH – Control collection of TCPIP Host details 89
JNM - Control the collection of job names 90
LDI – Set LOAD exclusion inactive 91
LDX – Add or activate a LOAD exclusion 91
LNK - Set linklist library collection status 91
LPA - Set link pack area program monitoring status 92
LPLN - Set the sysplex name 93
PRE - Collect usage for long running programs 93
PRI - Set the data set space primary allocation 94
PRS - Set registered software activity data collection status 94
QSZ – Specify collection element queue area size 94
REF - Refresh Usage Monitor settings 95
SEC - Set the data set space secondary allocation 95
SID - Set the Usage Monitor system identifier 96
SIZ - Set the data space repository size 96
SJS - Controlling spawned job suffix preservation 96
SWI - Switch to a new data space repository 97
TRCD-A - Display data set allocation parameters 103
TRCD-J - Display trace selection job name masks 103
TRCD-P - Display trace selection program name masks 103
TRCDCL - Set output data set SMS data class 103
TRCDSN - Set output data set name 103
TRCEXC - Control tracing of excluded events 104
TRCJOB - Specify job name trace selection filter 104
TRCJOD - Delete job name trace selection filter 105
TRCMCL - Set output data set SMS management class 105
TRCOFF - Terminate tracing 105
TRCON Initiate tracing 105
TRCPGD - Delete job name trace selection filter 106
TRCPGM - Specify program name trace selection filter 106
TRCPRI - Set output data set primary space 106
TRCSCL - Set output data set SMS storage class 106
TRCSEC - Set output data set secondary space 107
TRCSTA - Set output data set initial disposition 107
TRCUNIT – Set output data set device allocation unit 107
TRCUNX - Control tracing of UNIX syscalls 107
TRCVOL – Set output data set volume 108
UID - Control the collection of user details 97
UNK - Set the unknown event collection switch 98
UNM - Set user name collection status 98
UNT - Set the data set allocation unit 98
UNX - Set UNIX program monitoring status 99
VOL - Set the data set allocation volume 99
WRT- Set the automatic switch-and-write time of day 99
XDD - Deleting data set name exclusion entries 100
XDS - Adding data set name exclusion entries 100
ZIP - Set the compressed output data switch 100
Usage Monitor general commands 80
Usage Monitor messages, HSIZ 275
Usage Monitor trace commands 102
Usage Monitor Trace Facility 101
Usage Monitory
files used by 74
using exclusion masks to reduce data 75

V

variables, syntax diagrams 4
verifying database changes since the product was released 168–170

W

work pool
Inquisitor requests 63

X

XVOLUME 57

Z

z/OS UNIX security 13