IBM Data Virtualization Manager for z/OS
Version 1 Release 1

Administration Guide

SC27-9303-00

Contents

Figures

Tables

About this information

This information supports IBM Data Virtualization Manager for z/OS (5698-DVM) and contains information about the Data Virtualization Manager server, which is a component that is provided with IBM Data Virtualization Manager for z/OS.

Purpose of this information

This document presents the information you need to perform administrative tasks while using the Data Virtualization Manager server.

Who should read this information

This document is intended for system administrators, system programmers, and database administrators.

How to send your comments to IBM

We appreciate your input on this documentation. Please provide us with any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

Important: If your comment regards a technical problem, see instead "If you have a technical problem" on page xvii.

Send an email to comments@us.ibm.com.

Include the following information:

- Your name and address
- Your email address
- Your phone or fax number
- The publication title and order number:

  IBM Data Virtualization Manager for z/OS Administration Guide
  SC27-9303-00
- The topic and page number or URL of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM®, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are listed for sending comments. Instead, take one or more of the following actions:

- Visit the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Chapter 1. Overview

IBM Data Virtualization Manager for z/OS enables data from multiple, disconnected sources to be virtually integrated into a single, logical, data source and shared with any application, providing the right data, in the right format, at the right time.

This guide includes information on performing the following IBM Data Virtualization Manager for z/OS administrative features:

- Starting the ISPF application
- Virtualizing and accessing mainframe data
- Security
- Performance
- Configuring rules and events
- Logging and tracing server information
- Monitoring
- Managing users and system resources
- Distributed transactions
- Migrating maps
- SQL DMF supported data types

What's new in IBM Data Virtualization Manager for z/OS Administration Guide

This section describes recent technical changes to IBM Data Virtualization Manager for z/OS.

New and changed information is marked like this paragraph, with a vertical bar to the left of a change. Editorial changes that have no technical significance are not marked.

Description	Related APARs
The Integrated DRDA Facility (IDF) introduces a DRDA Application Server (AS) into Data Virtualization Manager, allowing for peer-to-peer communications between Data Virtualization Manager servers. Each Data Virtualization Manager server can use DRDA to access data sources resident at another peer Data Virtualization Manager server. See "Using multiple Data Virtualization Manager servers as peers" on page 294.	PH05549
The Data Virtualization Manager server can now listen for ENF 55 auxiliary storage shortage signals and throttle storage utilization when an auxiliary storage shortage is signaled. The point at which the Data Virtualization Manager server will reject new connection attempts when an auxiliary storage shortage is signaled by the system Event Notification Facility is controlled by the server parameter `DSCLIENTAUXSTGCUTOFF`. See "Modifying the client auxiliary storage cut-off parameter" on page 291.	PI97462
Information has been added about configuring security to access Adabas data. See "Configuring Adabas security" on page 110.	

Chapter 2. ISPF Application

ISPF is a multifaceted development tool set for the z/OS® operating system. Since 1975, MVS™ programmers have used ISPF for host-based application development productivity. ISPF forms the basis of many TSO and CMS applications and provides extensive programmer-oriented facilities as well. For more information on ISPF, refer:

- https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.4.0/com.ibm.zos.v2r4.f54ug00/intro.htm
- https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconc_whatisispf.htm

Subsystem Manager

The subsystem manager is a program that helps you to view the available subsystems, start/stop a subsystem, and navigate to IN000, job log or started task JCL of a subsystem from the **ISPF Command Shell**.

Procedure

To invoke the subsystem manager:

1. From the **ISPF Primary Option Menu**, enter the option **6**. The **ISPF Command Shell** window appears.

```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
   Menu  List  Mode  Functions  Utilities  Help

                              ISPF Command Shell
Enter TSO or Workstation commands below:

===>



Place cursor on choice and press enter to Retrieve command

=>
=>
=>
=>
=>
=>
=>
=>
=>
=>
```

2. Enter the command **dv** in the **ISPF Command Shell** to see the available subsystems. The following figure shows the available subsystems.

```
-------------------------------- DV Subsystems ------------- Row 1 to 12 of 36

PLEX: RSPLEX01  LPAR: RS28      MODE: EDIT     CONFIRM: YES

  LCs: A Admin  S Start  P Stop  J JCL  I IN00  O Output  D Details  F Files

   DVSS LPAR Jobname  Status ServCls
   ADBJ RS28 ADBJ     UP     SDB_SCNM
   ADBV RS28 ADBV     UP     SDB_SCHI
   AVZB RS28 AVZB     UP     AVZ_SCNM
   AVZC RS28 AVZC     UP     AVZ_SCHI
   AVZS RS28 AVZS     UP     AVZ_SCHI
   AVZY RS28 AVZY     UP     AVZ_SCHI
   AVZ1 RS28 AVZ1     UP     AVZ_SCHI
   AVZ3 RS28 AVZ3     UP     SDB_SCNM
   AVZ4 RS28 AVZ4     UP     AVZ_SCNM
   AVZ5 RS28 AVZ5     UP     AVZ_SCNM
   AVZ9 RS28 AVZ9     UP     AVZ_SCHI
   AZKS RS28 AZKS     UP     SDB_SCNM

Command ===>                                                  Scroll ===> CSR
```

The subsystem manager menu has the following options:

- **A Admin**: Opens the IBM Data Virtualization Manager for z/OS ISPF administration menu.
- **S Start**: Starts the subsystem.
- **P Stop**: Stops the subsystem.
- **J JCL**: Opens the server started Proc JCL file.
- **I IN00**: Opens the server's IN00 file.
- **O Output**: Opens the job log file.
- **D Details**: Displays the last modified details of the server such as parameter, IN00 and Proc files modification details.
- **F Files**: Shows the datasets allocated/running in the current server.

IBM Data Virtualization Manager for z/OS ISPF application

You can perform Data Virtualization Manager server administrative functions, data virtualization, and mainframe access by using the ISPF application.

You can use the ISPF application with or without the Instrumentation Server. The Instrumentation Server is a management environment that traces and integrates activity from all mainframe nodes in a sysplex and graphically displays the information in the Data Virtualization Manager studio.

If you are unfamiliar with the basic functionality of an ISPF application, see the online tutorial. To access the tutorial, start the ISPF application, type HELP on the command line, and press Enter.

Invoking the IBM Data Virtualization Manager for z/OS ISPF application using the Command Shell

You can invoke the Data Virtualization Manager server administrative menu by using the **ISPF Command Shell**.

Before you begin

- The Data Virtualization Manager server must be running.
- Your security administrator must give your TSO user ID READ, UPDATE, or both authorities. You need READ authority to view resource lists. You need UPDATE authority to modify server information.

Procedure

1. From the **ISPF Primary Option Menu**, enter the option **6** and press ENTER.
2. Enter the following command in the **ISPF Command Shell**:

- If you are only using the Data Virtualization Manager server:

```
'hlq.SAVZEXEC(AVZ)' 'SUB(AVZS)'
```

- If you are using the Instrumentation Server:

```
'hlq.SAVZEXEC(AVZ)' 'SUB(AVZS) TBSSID(AVZS)'
```

where `hlq` is your high-level qualifier, `SUB (AVZS)` specifies the subsystem name for the Data Virtualization Manager server, and `TBSSID(AVZS)` specifies the subsystem name for the Instrumentation Server.

Results

The **Primary Option Menu** panel is displayed. This panel specifies the ID of the subsystem to which you are connected and information about the product version. From this panel, you access all of the functionality in the product.

```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                    IBM Data Virtualization Manager for z/OS


   Interface Facilities:                                  SSID    : AVZS
    1  ACI              5  IDMS                           Version : 01.01.00
    2  Adabas           6  IMS                            Date    : 20/03/10
    3  CICS             7  VSAM/Sequential                Time    : 08:12
    4  DB2              8  DSSPUFI

   Data Virtualization Server Administration:
    A  Remote User      -  Manage Remote Users
    B  Server Trace     -  Server Trace Facility - SIS SSID: AVZS
    C  AVZ Admin.       -  Manage Data Virtualization Server
    D  Data Mapping     -  Data Mapping Facility
    E  Rules Mgmt.      -  Event Facility Management
    F  Monitor          -  Monitor Server Activity
    G  Streams          -  Streams Administration
    H  Services         -  Services Administration
    I  Instrumentation  -  Instrumentation Server Administration

 Option  ===>
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Invoking the IBM Data Virtualization Manager for z/OS ISPF application using the Subsystem Manager

You can use the subsystem manager to quickly navigate to the administration menu of Data Virtualization Manager server.

Procedure

To invoke the Data Virtualization Manager server using the subsystem manager:

1. Enter the command **dv** in the **ISPF Command Shell** to see the available subsystems.
2. Type **A** next to the desired subsystem to open the administrative menu of the subsystem.

   In the following example, the administrative menu of the subsystem **AVZ4** is invoked.

```
-------------------------------- DV Subsystems ------------- Row 1 to 12 of 36

PLEX: RSPLEX01  LPAR: RS28      MODE: EDIT     CONFIRM: YES

  LCs: A Admin  S Start  P Stop  J JCL  I IN00  O Output  D Details  F Files

   DVSS LPAR Jobname  Status ServCls
   ADBJ RS28 ADBJ     UP     SDB_SCNM
   ADBV RS28 ADBV     UP     SDB_SCHI
   AVZB RS28 AVZB     UP     AVZ_SCNM
   AVZC RS28 AVZC     UP     AVZ_SCHI
   AVZS RS28 AVZS     UP     AVZ_SCHI
   AVZY RS28 AVZY     UP     AVZ_SCHI
   AVZ1 RS28 AVZ1     UP     AVZ_SCHI
   AVZ3 RS28 AVZ3     UP     SDB_SCNM
 A AVZ4 RS28 AVZ4     UP     AVZ_SCNM
   AVZ5 RS28 AVZ5     UP     AVZ_SCNM
   AVZ9 RS28 AVZ9     UP     AVZ_SCHI
   AZKS RS28 AZKS     UP     SDB_SCNM

Command ===>                                              Scroll ===> CSR
```

The administrative menu of the subsystem **AVZ4** is displayed.

```
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                    IBM Data Virtualization Manager for z/OS


  Interface Facilities:                                  SSID    : AVZ4
   1  ACI              5  IDMS                           Version : 01.01.00
   2  Adabas           6  IMS                            Date    : 20/03/10
   3  CICS             7  VSAM/Sequential                Time    : 08:12
   4  DB2              8  DSSPUFI

  Data Virtualization Server Administration:
   A  Remote User      -  Manage Remote Users
   B  Server Trace     -  Server Trace Facility - SIS SSID: AVZS
   C  AVZ Admin.       -  Manage Data Virtualization Server
   D  Data Mapping     -  Data Mapping Facility
   E  Rules Mgmt.      -  Event Facility Management
   F  Monitor          -  Monitor Server Activity
   G  Streams          -  Streams Administration
   H  Services         -  Services Administration
   I  Instrumentation  -  Instrumentation Server Administration

Option  ===>
```

IBM Data Virtualization Manager for z/OS ISPF Primary Option Menu

The **Primary Option Menu** for Data Virtualization Manager server provides access to interface facilities and administrative functions.

```
                      IBM Data Virtualization Manager for z/OS


   Interface Facilities:                                  SSID    : xDBy
    1  ACI               5  IDMS                          Version : vv.rr.mm
    2  Adabas            6  IMS                           Date    : yy/mm/dd
    3  CICS              7  VSAM/Sequential               Time    : hh:mm
    4  DB2               8  DSSPUFI

   Data Virtualization Server Administration:
    A  Remote User       -  Manage Remote Users
    B  Server Trace      -  Server Trace Facility - SIS SSID: AVZ9
    C  AVZ Admin.        -  Manage Data Virtualization Server
    D  Data Mapping      -  Data Mapping Facility
    E  Rules Mgmt.       -  Event Facility Management
    F  Monitor           -  Monitor Server Activity
    G  Streams           -  Streams Administration
    H  Services          -  Services Administration
    I  Instrumentation   -  Instrumentation Server Administration


 Option  ===>
```

Type the number or letter that corresponds to the task that you want to perform.

The information in the upper right part of the panel includes the following information:

- SSID: The ID of the Data Virtualization Manager server subsystem to which you are connected.
- Version: The version, release, and maintenance number of Data Virtualization Manager server.
- Build: The build number or SVFX of the product.
- Date: The date this version and build originated.
- Time: The time this version and build originated.

Table 1 on page 7 describes the interfaces to various data sources.

Table 1. Primary Option Menu - Interface Facilities

Option	Description
ACI	The ACI API allows clients to connect to backend programs in remote TP environments. See "IBM Data Virtualization Manager for z/OS Interface for ACI" on page 31
Adabas	The Data Mapping facility (DMF) allows for the creation of Adabas data maps for Server. See "IBM Data Virtualization Manager for z/OS Interface for Adabas" on page 53
CICS	The Server CICS Control Facility allows you to view and control the Server CICS environment. See "IBM Data Virtualization Manager for z/OS Interface for CICS/TS" on page 54.
DB2	The Server Database Control application allows you to view and modify the product Server Database table. See "IBM Data Virtualization Manager for z/OS Interface for DB2" on page 62
IDMS	The Server IDMS Control Facility allows you to monitor and control your access to IDMS/DC and IDMS/DB. See
IMS	The Server IMS Control Facility allows you to monitor and control your access to IMS/TM and IMS/DB. See "IBM Data Virtualization Manager for z/OS Interface for IMS DB: support for DBCTL" on page 64

Table 1. Primary Option Menu - Interface Facilities (continued)

Option	Description
VSAM/Sequential	The Data Mapping facility (DMF) allows for the creation of VSAM/ Sequential data maps for Server. See "IBM Data Virtualization Manager for z/OS Interface for VSAM and Sequential files" on page 75.
DSSPUFI	Evaluate SQL statements and view the results.

Table 2 on page 8 describes the options in the primary ISPF menu.

Table 2. Primary Option Menu - Server Administration

Option	Description
Remote User	Manage Remote Users
Trace Browse	Trace Browse Facility
AVZ Admin	Manage Data Virtualization Manager server
Data Mapping	Manage the Data Mapping Facility (DMS). See "Using the Data Mapping Facility" on page 80
Rules Mgmt.	Event Facility Management. See "Using the Event Facility" on page 317
Monitor	Monitor Server Activity. See Chapter 8, "Monitoring," on page 273
Streams	Streams Administration
Services	Services Administration
Instrumentation	Instrumentation Server Administration. See "Instrumentation Server" on page 276.

Manage Remote User

The **Manage Remote User** program allows you to view current and cumulative information about users connected to the IBM Data Virtualization Manager for z/OS.

Procedure

The type of information that you can view using this program includes:

- General information about the user, the link, and the security status of the host.
- TCP/IP-specific information. If the link is not a TCP/IP link, the fields contain asterisks.
- Information related to SQL processing. You can follow the progress of an application as it issues SQL statements.
- Information about the amount of time that the application is using in processing and in communication.
- Information about data compression and the amount of data that the application has transmitted and received.
- Information about the amount of time the application has spent processing and transmitting data.

To view information about a remote user:

1. From the **Primary Option Menu** panel, enter **A** to choose the option **Remote User**.
2. Press **ENTER**. The **Remote User** panel appears.

```
------------------------------- Remote Users ---------------- Scr 1 Row 1 of 1
   LCs: A AuthIDs    C Cancel Thread  E Explain Codes  F Format  I Info
        K Kill User  P Print CB  S Show CB  T User Trace  U User Detail

  Host     LAN           Host             Link   Application
  USERID   USERID        Name             Type   Name                Note
  TS1111   msssm         blr-l-ms01       OTC/IP Studio
**End**
```

To know more information on the remote user connections, refer "Managing user connections" on page 310.

Manage Data Virtualization Manager server

You can use this program to view or modify vital Data Virtualization Manager server data including parameter and other statistical information.

Invoking the IBM Data Virtualization Manager for z/OS Administration Menu

From the **ISPF Primary Menu**, select **AVZ Admin** and press ENTER. The **Server Management Menu** is displayed.

```
                       Server Management Menu                      SSID: AVZ9
Option  ===>
                                                                  More:     +

   1 ISPF Session    - Display and modify ISPF/AVZ session parameters
   2 AVZ Parms       - Display and modify AVZ main task parameters
   3 AVZ Blocks      - Display formatted AVZ control blocks
   4 AVZ Stats       - Display AVZ product statistics
   5 AVZ Tokens      - Display and control product tokens
   6 AVZ Modules     - Display product module information
   7 AVZ Tasks       - Display product tasks
   8 AVZ IP Tree     - Display the IP address tree
   9 AVZ Prcs Blks   - Display the Cross Memory Process Blocks
  10 AVZ RPC         - RPC Control Facility
  11 AVZ Copies      - Display information about each copy of the product
  12 AVZ Storage     - Display virtual storage information
  13 Trace Archive   - Server Trace Archive Facility
  14 AVZ Group       - Display all remote users in a group
  15 NLS Tables      - Display National Language Support tables
  16 Link            - Display Link Tables
  17 RRS             - Display RRS Facilities
  18 SOM             - Display and control Security Optimization and Managemant
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Displaying and Modifying ISPF/AVZ Session Parameters

The ISPF Session program identifies the subsystem name that is associated with your ISPF session. The settings are saved in the current user's profile variable pool under the ISPF application. This option is only available through the Data Virtualization Manager server ISPF panels

Procedure

To invoke the ISPF Session Parameters option:

1. Select the option **C** for **AVZ Admin** from the Data Virtualization Manager server Primary Option menu.
2. Select option **1** for **ISPF Session** and press ENTER. The system displays the following panel.

```
-------------------------  ISPF Session Parameters  -------------------------

   Subsystem Name (SSID)  ===> AVZ9

   Status Information:
     Product version      ===> 01.01.00
     Build number         ===> 0000
     Build date           ===> 2017-09-29

   Server Trace Facility  ===> AVZ9   (SIS Subsystem ID)
```

3. To change the subsystem name or the trace browse facility subsystem name for the Instrumentation Server (SIS), type the new name over the existing name and press ENTER. The status information updates to reflect the information for the subsystem associated with your ISPF session.

Displaying Formatted Control Blocks

The Data Virtualization Manager server Internal Control Block program displays the contents of critical product controls. The information is formatted to show individual fields and is followed by a hexadecimal dump of the control block controls.

Procedure

Select the option **C AVZ Admin** and select **AVZ Blocks** from the **Server Management Menu**. The **Internal Control Blocks** screen appears.

```
----------------------------  Internal Control Blocks  ----------- Row 1 of 30
   LCs: D Display  F Format  P Print CB  S Show CB

  Block Control Block                      Virtual   Storage   Pr
  Name  Description                  ASID  Address   Length    Ky  Note
  AECB  Security Optimization Block  0191  7E5AE000  00005030  80
  BOST  Active Browse Status Block   0191  00001000  00000800  88
  CICO  CICS Control Area            0191  00008000  00002000  80
  CIEC  EXCI Control Area            0191  3D8B8000  00001000  80
  CMAS  Product ASVT block           0191  7F2E1000  000086C8  80
  HSST  DFHSM Raw Recall Statistics  0191  7F709000  00000100  80
  IDCO  IDMS Control Area            0191  3D8BA000  00001000  80
  IMCO  IMS Control Area             0191  0000A000  00001000  80
  IMDA  IMS/ODBA Control Block       0191  3D8BB000  00000ABC  88
  OPCK  Execution Checklist          0191  3CA40000  00000360  00
  OPML  Message Lookup Table         0191  7F70B000  00018000  40
  OPMS  Product Master Block         0191  32B05000  0000A000  40
  OPPA  Product Parameter Table      0191  3CE79000  00045CA0  00
  OPPM  Permanent Data Area          0191  2D0F1000  00003000  40
  OPPT  Protected Data Area          0191  7F427000  00001000  40
  OPVN  Product Vendor Table Entry   0191  37556D58  00000200  40
Command ===>                                                Scroll ===> PAGE
```

The program supports the following line commands:

- **D**: Displays the product control block specified by the selected row.
- **F**: Formats the block selection entry for the selected row.
- **P**: Prints the block selection entry for the selected row.
- **S**: Displays the block selection entry for the selected row.

Type the command to the left of the line and press ENTER. In the following example, formatted information of CICS Control Area (CICO) is displayed.

```
 BROWSE    Internal Control Block Information       Line 0000000000 Col 001 053
********************************* Top of Data **********************************
Block Name (Acronym)                 CICO
Internal Entity Name                 CICO
Virtual Storage Address              X'00008000'
Storage Size                         X'00002000'
Block's ASID                         X'0191'
Virtual Storage Protect Key          X'80'
Short Desciption                     CICS Control Area
Desciption                           CICS Control Area
******************************** Bottom of Data ********************************
```

You can sort the columns according to their sort names. The describes the sort names of the columns available.

Table 3. Sort Names

Column Name	Description	Sort Name
BLOCK NAME	The name of the control block.	NAME
ASID	The ASID where the control block resides.	ASID
VIRTUAL ADDRESS	The virtual address of block storage.	ADDRESS
STORAGE LENGTH	The virtual size of block storage.	SIZE
PR KEY	The storage protection key.	KEY

Data Virtualization Manager server Statistics

Data Virtualization Manager server provides the following statistics that help to diagnose problems.

- **Task Process Blocks**. These blocks contain information needed to represent a dispatchable unit of work in the Data Virtualization Manager server. The blocks also contain space for dynamic save and work areas that are referred to as the stack. The size of the process block depends on the Data Virtualization Manager server features that are licensed.
- **ESTAE Routine Process Blocks.** End-of-task statistics are counts of the number of times the subsystem interface (SSI) was entered for the termination of all tasks and address spaces.
- **ABEND and LOGREC Statistics**. These statistics provide a summary of how often and how many abends are occurring.

Invoking Data Virtualization Manager server Statistics

Perform the following steps to invoke the statistics option.

Procedure

Select the option **C AVZ Admin** and select **AVZ Stats** from the **Server Management Menu**. The **Product Statistics** screen appears.

```
----------------------------  Product Statistics  -------------------------
Command ===>
                                                                   More:
     Task process blocks
        Number of process blocks in use .........                 1
        Number of attached subtasks .............                 8

     PC routines process blocks
        Total PC process block allocations ......             50897
        Blocks in use ..........................                  0
        Blocks recaptured ......................                  0
        High water clock value .................     28APR 22:41:42.224980
        Last allocation failure clock value .....
        Allocation failure count ...............                  0
        High water count .......................                  7

     ESTAE routine process blocks
        Total ES process blocks allocations .....                 0
        Blocks in use ..........................                  0
        Blocks recaptured ......................                  0
        High water clock value .................
        Last allocation failure clock value .....
```

Displaying and Managing Tokens
The Token Control program allows you to view and manage Data Virtualization Manager server execution tokens. This program is used to determine the status of a token, view token data, and kill tokens.

Procedure

Select the option **C AVZ Admin** and select **AVZ Tokens** from the **Server Management Menu**. The **Token control** screen appears.

```
------------------------------  Token Control  ------------- Scr 1 Row 1 of 0
   LCs: D Display Data  F Format  K Kill Token  P Print CB  S Show CB

  Token                    User      Host               Data
  ID                       ID        Name               Size  Note
**End**
```

The program also supports the following line commands:

- **D**: Displays the data associated with the token.
- **F**: Formats the data for the selected row.
- **K** Kills (deletes) the token and frees the associated data.
- **P**: Prints the token control block for the selected row.
- **S**: Displays the token control block for the selected row.

The following table describes each column name on the ISPF panel and provides the sort name (if available).

Table 4. Sort Names For Tokens

Column Name	Description	Sort Name
TOKEN ID	The hexadecimal string that uniquely identifies the token.	ID
USER ID	The client user ID that created the token.	USER
HOST NAME	The host name of the client that created the token. This field typically contains the IP address of the client in dot notation.	HOST

Table 4. Sort Names For Tokens (continued)

Column Name	Description	Sort Name
DATA SIZE	The size of the data area associated with the token.	SIZE
CREATION TIME	The GMT time when the token and the data area associated with the token were created.	CREATION
LAST USED	The GMT time when the token was last used. Any use of token (access, update) resets the date and time values.	LAST
TIMEOUT	The number of seconds when token can go unused before it is timed out. Note that this is always from the last use of any kind.	TIMEOUT
USER DATA	The user data that was specified when the token was created. This is not the same as the data associated with the token.	DATA

Product Tasks

The tasks program provides current and cumulative information about Data Virtualization Manager server. This section describes on how to invoke the product tasks program and the available commands on the program.

Invoking Product Tasks

To invoke the product tasks option:

Procedure

1. Select the option **C** for **AVZ Admin** from the Data Virtualization Manager server Primary Option menu.
2. Select option **7** for **AVZ Tasks** and press ENTER. The system displays the following panel.

```
------------------------------- Tasks --------------------- Scr 1 Row 1 of 100
   LCs: C Cancel Thread  F Format  K Kill Task  P Print CB  S Show CB
        T SQL Trace      U User Detail

  TCB       Connection  Execution  Program    CPU
  Address   ID          State      Name       Time      Note
  008BDBA0  001332FD    Running    TRACE      121.554S
  008BD6F0  001332FF    Running    OPCIEC     004.861S
  008A4218  00133313    Running    ZCPMGR     004.107S
  00899AE8  00133318    Running    OPCKLM     002.705S
  008B6850  00133303    Running    OPIMSR     .735354S
  008B2220  0013330A    Running    GLVA       .607014S
  008B6220  00133305    Running    OPOACN     .549330S
  008BD3D8  00133302    Running    OPRRRM     .538958S
  008A5A60  00133310    Running    SEFFULL    .300114S
  008A4080  00133315    Running    OPMAXM     .179258S
  008995C8  00133317    Running    OPSGIN     .172620S
  008BD888  001332FE    Running    OPMAEF     .167719S
  0088FAF0  0013333C    Running    OPMAEC     .072843S
  0086D128  00133342    Running    OPMAEC     .072394S
  008F71A0  001332FC    Running    OPINAS     .056643S
Command ===>                                                  Scroll ===> PAGE
```

Line Commands and Column Names

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents. It also supports the primary SORT and LOCATE commands.

The program also supports the following line commands:

Table 5. Line Commands

Line Commands	Description
C	Cancels the thread.
F	Formats the information for the selected row.
K	Kills the selected task.
P	Prints the associated control block for the selected row.
S	Displays the control block for the selected row.
T	Displays an SQL trace for the selected task.
U	Displays user detail for the selected row.

Type the command to the left of the line and press ENTER.

The following table describes each column name on the ISPF panel and provides a sort name (if available).

Table 6. Column Names

Column Name	Description	Sort Name
TCB ADDRESS	The name of the product module.	TCB
CONNECTION ID	The address of the product module.	CONNECTION
EXECUTION STATE	The status of subtasks.	
PROGRAM NAME	The product module size.	PROGRAM
CPU TIME	The amount of CPU time used by the TCB.	CPU
SMAF ADDRESS	The address of the SMAF control block.	SMAF
TASK TYPE	A description of the type of task.	TASK

Product Copies

The product control program provides information about different copies of Data Virtualization Manager server in use. This program is view-only.

Invoking Product Copies

Procedure

To invoke product copies program:

1. Select the option **C** for **AVZ Admin** from the Data Virtualization Manager server Primary Option menu.
2. Select option **11** for **AVZ Copies** and press ENTER. The system displays the following panel.

```
------------------------------  Product Control  ----------- Scr 1 Row 1 of 15
   LCs: F Format  P Print CB  S Show CB

  Product  Group     Product   Network   Transfer  Current
  Name     Name      Status    Type      Status      Users   Note
  AVZG     None      Up        OTC/IP    Enabled         0
  AVZM     None      Up        OTC/IP    Enabled         0
  AVZS     None      Up        OTC/IP    Enabled         0
  AVZU     None      Down      None      None            0
  AVZW     None      Up        OTC/IP    Enabled         0
  AVZY     None      Up        OTC/IP    Enabled         0
  AVZ1     None      Up        OTC/IP    Enabled         0
  AVZ2     None      Up        OTC/IP    Enabled         0
  AVZ3     None      Up        OTC/IP    Enabled         3
  AVZ4     None      Up        OTC/IP    Enabled         0
  AVZ5     None      Up        OTC/IP    Enabled         0
  AVZ6     None      Up        OTC/IP    Enabled         0
  AVZ8     None      Down      OTC/IP    Disabled        0
  AVZ9     None      Up        OTC/IP    Enabled         1
  XVZX     None      Up        OTC/IP    Enabled         0
**End**
Command ===>                                                  Scroll ===> PAGE
```

Commands and Column Names

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PFkey equivalents or scroll bar equivalents. It also supports the primary SORT and LOCATE commands.

The program also supports the following line commands:

Table 7. Line Commands

Line Commands	**Description**
F	Formats the information for the selected item.
P	Prints the control block information related to the selected item.
S	Displays the control block information related to the selected item.

Type the command to the left of the line and press ENTER.

The following table describes each column name on the ISPF panel and provides a sort name (if available).

Table 8. Column Names

Column Name	**Description**	**Sort Name**
PRODUCT NAME	The 4-character name of the server subsystem (started task) with which this ISPF session is to communicate.	PRODUCT
GROUP NAME	The name of the load balancing group.	GROUP
PRODUCT STATUS	The status of the server (for example, up or down).	STATUS
NETWORK TYPE	The type of network on which the Server is running (for example, TCP/IP or SNA using LU6.2 protocol).	NETWORK

Table 8. Column Names (continued)

Column Name	Description	Sort Name
TRANSFER STATUS	The status of transfer.	
CURRENT USERS	The number of current users of the server.	
PRODUCT VERSION	The product version and release number.	
HI-WATER USERS	The highest number of users connected concurrently.	
LICENSED MAXIMUM	The maximum number of users allowed by license.	
INSTALLATION MAXIMUM	The maximum number of users allowed.	
UNALLOC<16 MEG	The size of the free storage in the blocks of virtual storage allocated to this server that reside below the 16 MB line.	
UNALLOC>16 MEG	The size of the free storage in the blocks of virtual storage allocated to this server that reside above the 16 MB line.	
MINIMUM <16 MEG	The lowest size of the owned blocks of virtual storage allocated to this server that reside above the 16 MB line minus the amount of free storage in those blocks.	
MINIMUM>16 MEG	The lowest size of the owned blocks of virtual storage allocated to this server that reside above the 16 MB line minus the amount of free storage in those blocks.	
ALLOCATED<16 MEG	The size of the owned blocks of virtual storage allocated to this server that reside below the 16 MB line.	
ALLOCATED>16 MEG	The size of the owned blocks of virtual storage allocated to this server that reside above the 16 MB line.	
SSCT ADDRESS	The address for the subsystem control block.	
SSVT ADDRESS	The address for the subsystem vector table.	
OPMS ADDRESS	The address for the main product control block.	

Table 8. Column Names (continued)

Column Name	Description	Sort Name
OPPM ADDRESS	The address for the product permanent data area.	
ASID VALUE	The MVS address space index.	

General ISPF Guidelines

This sections describes basic navigating guidelines for the Data Virtualization Manager server's ISPF application.

The Data Virtualization Manager server's ISPF application can be used with or without the Instrumentation Server (SIS). The SIS is a management environment that traces and integrates activity from all mainframe nodes in a sysplex and graphically displays the information using Data Virtualization Manager server Studio.

Types of Commands

When working with the Data Virtualization Manager server's ISPF application, you can use the following types of commands:

- Display commands
- System control commands
- Primary commands

Display Commands

The display commands controls the display of data. For example, the UP and DOWN scrolling commands are display commands.

System Control Commands

The system control commands that are application specific change the system’s operating status.

Primary Commands

All panels have an Option or Command field.

```
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                       Server Management Menu                       SSID: AVZ3
Option  ===>
                                                                   More:     +
```

You can enter primary commands in this field by typing the command name (also known as the command verb) and pressing ENTER.

You can issue ISPF built-in commands from the command field of any ISPF panel. The most commonly used primary commands are:

- **HELP**: Invokes the online tutorial. The help is context-sensitive. For example, if you are in the Link program, invoking this command displays information about controlling links.
- **END**: Causes the current display to be abandoned, and returns you to the previous panel. It is also used to terminate the tutorial.
- **RETURN**: Returns control to the Data Virtualization Manager server's **Primary Option Menu**.
- **SPLIT**: Causes the display to be split into two logical displays. The split occurs on the line the cursor is currently positioned.

- **KEYS**: Displays the current PF key settings and allows you to change them.
- **PFSHOW**: Displays the current PF key settings at the bottom of the panel. You cannot modify PF key settings using **PFSHOW** command.
- **PRINT**: Records the current panel image in the ISPF list file, which can later be printed.

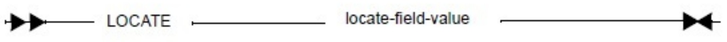
To associate a command to a PF key:

1. Enter the operand in the Option field. Examples of operands include UP, DOWN, and SPLIT commands.
2. Press the PF key you want associated with the command. ISPF automatically concatenates the command verb with the operands and simulates the pressing of the ENTER key.

Locate Data

The LOCATE command finds and scrolls the display to a specified row. Once a display is sorted on a particular column, the column becomes the search column for the LOCATE command.

The syntax of the LOCATE command is:



The **`locate-field-value`** indicates the row you want to scroll. This value must be in the same format as the data in the sort column. For example, if the sorted field is a decimal number, **`locate-field-value`** must also be a decimal number. For character strings, you do not need to specify a string that is the full length of the column. If you specify a shorter string, the LOCATE command pads locate-field-value with blanks to the length of the field.

Sort Data

Some scrollable programs support sorting and locating data. The SORT command is a primary command that sorts the columns of a display. The syntax of the SORT command is:

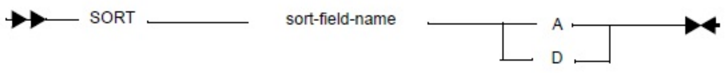


Table 9. Sort ISPF Data

Parameter	Description
`sort-field-name`	1- to 8-character identifier for the column to be sorted.
A	Indicates that the column is sorted in an ascending sequence (smallest to largest).
D	Indicates that the column is sorted in a descending sequence (largest to smallest).

Splitting the Screen

Using the SPLIT primary command, you can split your ISPF session into two logical sessions. Only one session is active at a time. The active session is the one that contains the cursor. To move between sessions, use the SWAP command.

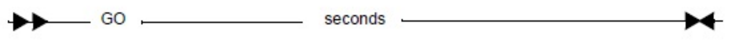
Alternatively, if a part of the inactive window is visible, you can simply move the cursor to the inactive window to move to that session.

To terminate a session, you can exit either by backing out through the session's primary option menu or by using the **=X** jump function.

Auto-Refresh

Some ISPF programs support the GO command. The GO command places the display in an auto-refresh mode. When a display is in auto-refresh mode, the keyboard is locked and the program periodically simulates an ENTER action.

The syntax of the GO command is:



where seconds is a value from 1 to 60, indicating the amount of time that the program waits between refresh cycles. To terminate auto-refresh mode, use the attention key (or PA1 on some terminals).

Note: If two attention actions are performed one after the other, it can cause the program to terminate.

Getting Help

To access the online tutorial, type the HELP command and press ENTER, or press the HELP. PF key from any panel of the application (the HELP command is usually assigned to F1, but that can be changed).

To end the tutorial and return to the Data Virtualization Manager server's ISPF application, type the END command or press the END PF key (usually F3).

Chapter 3. Virtualizing and accessing mainframe data

You can virtualize and access mainframe data using batch processing, the ISPF Server Data Mapping Facility, or the Data Virtualization Manager studio.

- *Batch* is typically used in a production lifecycle for adding and updating maps in your production environment. Batch provides an audit trail for monitoring mainframe changes.
- The *Data Virtualization Manager server ISPF* interface provides interface facilities for accessing data sources and a Data Mapping Facility for creating maps.
- The *Data Virtualization Manager studio* allows you to connect to data sources and map data. In Data Virtualization Manager studio data maps are referred to as virtual tables and virtual collections. For more information, see the *IBM Data Virtualization Manager for z/OS User's Guide*.

Virtual tables (maps)

Mapping data means that the source data's definition is used to create a virtual table that matches the definition of the source data. In the Data Virtualization Manager studio data maps are referred to as virtual tables for SQL solutions or virtual collections for NoSQL solutions.

The data definition depends on the programming language that compiles it. For example:

- For COBOL, it is a file definition or data definition.
- For PL/I, it is a Data Control Language (DCL) statement.
- For C, it is a structure.
- For Assembler, it is a DSECT instruction.

The information (length, format, and field elements) is extracted from the data definition and made available to the Data Virtualization Manager server. The data maps refresh process is governed by the **`Auto Refresh`** parameter that is specified by using the `Data Mapping Defaults Options`. To access this feature in ISPF, select `D Data Mapping` from the **Primary Option Menu**. Select `0 Map Defaults` from the **Server Data Mapping Facility Menu** and then set the **`Auto Refresh`** parameter to `Yes`.

Once created, a data map is called by using a parameter that is passed with an ODBC/JDBC SQL statement. The data map controls the parsing and formatting of the result set, including the names that are assigned to columns. By calling different maps, the Data Mapping Facility (DMF) can return different views or subsets of the data.

For data sources that you access by using a custom CALL-based RPC, you can use a data map to generate a skeleton RPC in COBOL. The skeleton contains the row-parsing code. You add application logic to the skeleton to produce the final RPC.

Data maps are created by using a series of ISPF panels that allow you to specify a data set containing a compile listing of a program that contains a data definition. The information (length, format, type, and offset, for example) about each field element is extracted from the data definition and made available to Data Virtualization Manager server.

Applications that use Data Virtualization Manager server through a Data Virtualization Manager client, JDBC, and ODBC can use the data maps to manipulate or view the logical or physical data.

Note: The extracts for COBOL and PLI data maps are also available in batch. The AVZMFPAR member is included in the server distributed *hlq*.SAVZCNTL data set as a sample JCL for extracting these types of maps in batch.

When you use the DMF, follow these guidelines:

- Use one server as a test server and another server as a production server.
- Use the DD statement AVZMAPP as part of your initial setup to identify the data sets that contain the maps for your production server.

- For each server, allocate one or more data sets, as needed. To facilitate central control of the production map data set, allocate a “staging” data set for interim maps.

Restrictions for non-supported clauses

Data mapping does not support OCCURS clauses that contain a DEPENDING ON clause.

When the OCCURS clause is used, it appends a numeric suffix to the corresponding column. For example, suppose you used the OCCURS clause that follows on the FIELD-A:

```
05 FIELD-A occurs 3 times
```

The result would contain the following column names:

```
FIELD-A-1
FIELD-A-2
FIELD-A-3
```

Restrictions for column extraction

The DMF can process up to 7,500 columns for a result set. If more than 7,500 columns are extracted, the extract process continues, but it is recommended that you disable any unwanted columns to reduce the total to 7,500.

ACF2 security checking

If your environment uses the ACF2 security product, authorization for data map imports is performed by using the user ID of the user that performs the import rather than the user ID of the Data Virtualization Manager server address space.

Using batch JCL jobs to create or copy maps

Batch processing is typically used in a production lifecycle for adding and updating maps in your production environment. Batch JCL jobs provide an audit trail for monitoring mainframe changes.

There are different ways to use batch processing:

- You can create a virtual table in the Data Virtualization Manager studio and use a batch job to copy the map into your production environment.
- You can use the batch job to create the new virtual table that is then put directly into your production environment.

Note: Using the Data Virtualization Manager studio is the recommended method to create virtual tables.

Extracting maps via batch jobs

You can extract maps for COBOL, PLI, Sequential Files, DBD, PSBs, ADABAS, and VSAM data sources, as well as for MFS maps and stored procedures. Use a sample batch job in member AVZMFPAR located in your *hlq*.SAVZCNTL data set for extracting these maps in batch.

Procedure

You still must use a compiled listing to perform the extract.

A COBOL listing with OPT(FULL) cannot be processed to produce a map. Keywords for this process define the same elements that you would specify on the ISPF panels. Batch extract does not support alternate indexes for VSAM.

See “The batch extract member” on page 23 for a description of parameters you can set for batch processing.

The batch extract member

You can use the AVZMFPAR member to extract batch maps for COBOL, PLI, Natural, Sequential Files, DBD, PSBs, ADABAS, CICS, and VSAM data.

Note: You must perform a mapping refresh before it shows in the display map command.

Table 10 on page 23 through Table 20 on page 27 describe the parameters that can be used in the AVZMFPAR member.

Table 10. To Fingerprint a File

Required?	Parameter	Description
Required	SSID = AVZS	The target subsystem to use this map.
Required	FUNCTION = FRPT	The function to be performed by the DMF parser. Fingerprint (FRPT). When a file is fingerprinted, the file is scanned to attempt to determine the language type, such as COBOL.
Optional	SOURCE =	The name of the data set that contains the source to parse. **Note:** It is recommend that, instead of using this parameter, you use the //SOURCE DD statement, which overrides this parameter.

Table 11. Source to DMF

Required?	Parameter	Description
Required	SSID = AVZS	The target IBM Data Virtualization Manager for z/OS subsystem to use this map.
Required	FUNCTION = STOD	The function to be performed by the DMF parser.
Optional	SOURCE = HLQ.SOURCE.FILE	The name of the data set that contains the source to parse. **Note:** It is recommended that, instead of using this parameter, you use the //SOURCE DD statement, which overrides this parameter.
Required	START FIELD =	The name of the first field to map.
Optional	END FIELD =	The name of the last field to map.
Optional	OFFSET ZERO = Y/N	Specifies whether to set the Start Search Field offset to zero, even if it is not a group level or the first definition in a group. Defaults to YES.
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.

Table 11. Source to DMF (continued)

Required?	Parameter	Description
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH (default) • REFRESH

Table 12. Source to DMF - Sequential

Required?	Parameter	Description
Required	SEQ FILE =	Specifies the data set to associate with the map (implies a sequential map for use by the sequential interface).
Optional	SEQ DSN COLUMN NAME =	The sequential request data set column name, if data set name (for PDS(E) data sets), that can be viewed by the client.
Optional	SEQ MEMBER COLUMN NAME =	The sequential request member column name, if member name (for PDS(E) data sets) that can be viewed by the client.
Optional	SEQ COLUMN NAME SEARCHABLE =	The sequential request DSN and member column names that can be used on the WHERE clause of a SQL statement.

Table 13. Source to DMF - To merge Map B into Map A

Required?	Parameter	Description
Required	SSID = AVZS	The server subsystem that uses this map.
Required	FUNCTION = MMER	The function to be performed by the DMF parser.
Required	MERGE A =	The map that contains the merged information (Map A of a merge function).
Required	MERGE B =	The name of the map that is merged into Map A.
Optional	NEW MAP NAME =	The name of the new map (structure name). The maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH (default) • REFRESH

Table 14. Source to DMF - To merge a map into a DBD segment

Required?	**Parameter**	**Description**
Required	SSID = AVZS	The target server subsystem to use this map.
Required	FUNCTION = MDBD	The function to be performed by the DMF parser.
Required	DBDNAME =	The name of the DBD to link to or unlink from.
Required	SEGMENT =	The name of the segment in the DBDNAME to link to or unlink from.
Required	LINK MAP =	The name of the map to link to the segment.
Optional	SAVE OPTION =	Specifies The DMF import save option. Valid values are: • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH (default) • REFRESH
Optional	DISABLE DUP = Y/N	Indicates whether to disable duplicates in the DBD.
Optional	DISABLE FILLER = Y/N	Indicates whether to disable filler fields in the DBD.

Table 15. Source to DMF - To remove a map from a DBD segment

Required?	**Parameter**	**Description**
Required	SSID = AVZS	The target server subsystem to use this map.
Required	FUNCTION = MDBD	The function to be performed by the DMF parser.
Required	DBDNAME =	The name of the DBD to link to or unlink from.
Required	SEGMENT =	The name of the segment in the DBDNAME to link to or unlink from.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH (default) • REFRESH

Table 16. Source to DMF - To convert a map to a sequential map

Required?	**Parameter**	**Description**
Required	SSID = AVZS	The target subsystem to use this map.
Required	FUNCTION = MTOS	The function that the parser performs.
Required	INPUT MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Required	SEQ FILE =	The data set associated with the map (implies a sequential map for use by the sequential interface).
Optional	SEQ DSN COLUMN NAME =	The sequential request data set column name, if data set name (for PDS(E) data sets), that can be viewed by the client.
Optional	SEQ MEMBER COLUMN NAME =	The sequential request member column name, if member name (for PDS(E) data sets) that can be viewed by the client.
Optional	SEQ COLUMN NAME SEARCHABLE =	The sequential request DSN and member column names that can be used on the WHERE clause of a SQL statement.
Optional	NEW MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH (default) • REFRESH

Table 17. VSAM from Source

Required?	**Parameter**	**Description**
Required	VSAM FILE = HLQ.VSAM.FILE	The VSAM file to be associated with this map (implies a VSAM map for use by the VSAM or CICS VSAM interface).
Optional	ALT INDEX = Y/N	Indicates that you want to use alternate indexes to access this VSAM map. Default is NO.
Optional	NEW MAP NAME =	The name of this map, which is known as the structure name. Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.

Table 18. To Convert a Map to a VSAM Map

Required?	Parameter	Description
Required	SSID = AVZS	The target server subsystem to use this map.
Required	FUNCTION = MTOV	The function to be performed by the DMF parser.
Required	INPUT MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.
Required	VSAM FILE = HLQ.VSAM.FILE	The VSAM file to be associated with this map (implies a VSAM map for use by the VSAM or CICS VSAM interface).
Optional	ALT INDEX = Y/N	Indicates whether to use alternate indexes to access this VSAM map. Default is `NO`.
Optional	NEW MAP NAME =	The name of this map (structure name). Maximum length is 30 bytes. This field is ignored for maps that require specific names such as the DBD and PSB maps. Defaults to the start field structure name.

Table 19. CICS

Required?	Parameter	Description
Required	CICS CONN =	The name of the CICS connection to use for this map, if this map is to be used by the CICS VSAM interface.
Required	CICS TRAN =	The name of the CICS transaction to use for this map, if this map is to be used by the CICS VSAM interface.
Required	CICS FCT = or CICS FCT ENTRY =	The name of the CICS FCT entry to use for this map, if this map is to be used by the CICS VSAM interface.
Optional	AIX*n* FCT = where *n* is numeric for 1-8.	The name of the CICS FCT entry to use for each IX path found, if this name is to be used by the CICS VSAM interface.
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: • NOSAVE • SAVE (default) • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH (default) • REFRESH

Table 20. Adabas - Supported Input Parameters for Extracting an Adabas File

Required?	Parameter	Description
Required	SSID = AVZS	The Data Virtualization Manager subsystem ID.
Required	FUNCTION = ADLF	The parser function for Adabas.

Table 20. Adabas - Supported Input Parameters for Extracting an Adabas File (continued)

Required?	Parameter	Description
Required	MAP NAME =	The name of this map, which is known as the structure name. The maximum length is 30 bytes.
Required	ADABAS DBID =	The database ID as shown on the ADAREP.
Required	ADABAS DBNAME =	The database name as shown on the ADAREP. This name is used for reporting purposes.
Required	ADABAS FILE NUM =	The file number of the Adabas file as shown on the ADAREP.
Required	ADABAS FILE NAME =	The SQL table name for the Adabas file.
Required	ADABAS SUBSYS =	The Adabas router name assignment. If not specified, the default is ADAB.
Optional	MU COUNT =	The maximum allowed MU (multiple value field) columns generated. If not specified, the default is 0.
Optional	PE COUNT =	The maximum allowed PE (periodic groups) columns generated. If not specified, the default is 0.
Optional	CREATE COUNT FIELDS =	If set, the parser generates a count field for all MU and PE fields. The name that is generated for the field is the PE or MU field name plus the letters "_C" (if using the field name in the DDM) or the PE or MU field name plus the letter "C" (if using the field name from the LF command). For example, if you run SDDMBTPA by using the DDM, and the PE or MU name ACCOUNTS, the generated name for the count field is ACCOUNTS_C. If you run SDDMBTPA by using only the LF command, and the PE or MU name is AA, the generated name for the count field would be AAC. Values are: • Y for Yes • N for No
Optional	U_2_P =	Indicates whether the extract converts all unpacked format fields to the packed format. Values are: • Y for Yes • N for No The default is N. **Note:** Use this parameter if you anticipate negative Adabas unpacked decimal numbers; otherwise, an alphanumeric representation is returned. For example, -23 would be returned as 02L. Use of this parameter changes the data type from character to numeric.

Table 20. Adabas - Supported Input Parameters for Extracting an Adabas File (continued)

Required?	Parameter	Description
Optional	B_2_I =	Indicates whether the extract converts all 2-byte and 4-byte binary file fields to short integer and integer formats respectively. Values are: • Y for Yes • N for No The default is N.
Optional	DE SEARCH ONLY =	Generates control definitions that allow the client to use WHERE columns that are Adabas descriptors (such as SUPERDE, SUBDE, and HYPERDE). Values are: • Y for Yes • N for No The default is N.
Optional	SEARCH BY PE INDEX =	Allows the client to target rows that match a particular occurrence of the PE field when searching rows by using the WHERE clause. If not specified, all rows where any occurrence of that PE field matches the value specified are targeted. Values are: • Y for Yes • N for No The default is N.

Optional	USE DDM =	Uses the DDM source supplied on the source DD statement to update the Adabas map with long field names and to override the data types as defined in the Adabas FDT. The DDM must be extracted using step DDMEXTR in this JCL.
		
		To generate a DDM member, execute a Natural batch job. For example:
		
		`//CMPRINT DD DISP=SHR,DSN=hlq.DDMS(DDMEMBER)`
		`//CMWKF01 DD DUMMY`
		`//CMSYNIN DD *`
		`    LOGON LIB`
		`    LIST VIEW ADABAS-DDM FIN`
		
		Valid values are:
		• Y for Yes
		• N for No
		
		The default is N.

Table 20. Adabas - Supported Input Parameters for Extracting an Adabas File (continued)

Required?	Parameter	Description
Optional	SAVE OPTION =	Specifies the DMF import save option. Valid values are: • NOSAVE • SAVE • REPLACE It is recommended that you use the SAVE value to prevent overwriting another map. The default is SAVE.
Optional	REFRESH OPTION =	Specifies whether to refresh the map. Valid values are: • NOREFRESH • REFRESH The default is NOREFRESH.
Optional	SECURITY =	Generates security on the "TABLE DEFINITION". Values are: • Y for Yes • N for No The default is N. To define a Data Virtualization Manager Resources for Adabas file security, you must edit and submit one of the following sample jobs (depending on your security type) located in the *hlq*.SAVZCNTL library: • AVZRAVDA for RACF security • AVZAZVDA for CA ACF2 security • AVZTSVDA for CA Top Secret Security
Optional	ADASCRPWD =	The password that is used to access the specified file number. If the IBM Data Virtualization Manager for z/OS Interface for Adabas accepts the password, it passes it to the **Adabas control block ADDS 3** field and generates the `ADASCRPWD =password` statement.
Optional	DBCS =	Specifies which Adabas Alpha/Binary to use to store pure DBCS data without SO/SI characters.

Table 21. Adabas - Redefine Parameters

Required?	Parameter	Description
Optional	REDEFINE_FORMAT = *x*	The 1-byte format type to be redefined. The rules for redefining a field format must conform to the rules of data type conversions that Adabas permits; otherwise, an Adabas response code might be generated because of a conversion mismatch.
Optional	REDEFINE_LENGTH = *nnn*	The length override.

Table 21. Adabas - Redefine Parameters (continued)

Required?	Parameter	Description
Optional	REDEFINE_COLUMN = *xxxxxxx...*	The 30-character name for the new redefined field that replaces the elements that comprise the original field. For example, if you are redefining field AA as two new fields or columns, the REDEFINE_COLUMN would indicate the new names for the two new fields: AA_PART_1 and AA_PART_2.
Optional	REDEFINE_OFFSET = *nnn*	The offset of the new redefined field, where *nnn* is the redefined offset to use.
Optional	REDEFINE_AS_COUNT	This option is used to support the SELECT COUNT(*) statement when there is no unique descriptor (DE, UQ) or fixed-format descriptor (DE, FI).
Optional	SET_AS_PRIMARYKEY	Allows you to set the field that is used as the primary key when there is no unique descriptor (DE, UQ).

Using the ISPF application to create or copy maps

IBM Data Virtualization Manager for z/OS supports access to many data sources.

- IBM Data Virtualization Manager for z/OS Interface for ACI
- IBM Data Virtualization Manager for z/OS Interface for Adabas
- IBM Data Virtualization Manager for z/OS Interface for CICS/TS
- IBM Data Virtualization Manager for z/OS Interface for DB2
- IBM Data Virtualization Manager for z/OS Interface for IDMS Control Facility
- IBM Data Virtualization Manager for z/OS Interface for IMS DB: Support for DBCTL and ODBA
- IBM Data Virtualization Manager for z/OS Interface for VSAM and Sequential Files

IBM Data Virtualization Manager for z/OS Interface for ACI

The Advanced Communication Interface (ACI) enables applications that are written in COBOL, Assember, PL/I, or Natural and running in remote transaction processing (TP) environments to communicate with the desktop.

ACI allows developers to create applications that can run services in their transaction processing (TP) environments. In this case, the IBM Data Virtualization Manager for z/OS Interface for ACI provides access to transactions in a CICS or Batch environment using the ACI API.

This interface also provides data access to JDBC and ODBC clients, web browser clients, and *n*-tier applications. It allows any JDBC- or ODBC-enabled application to use standard JDBC or ODBC facilities to make requests directly to a COBOL, Assembler, PL/I, or Natural program. A relational result set is returned to the application running in its native transaction processing environment.

The ACI Interface Facilities option on the Data Virtualization Manager server - Primary Option Menu provides access to the Server ACI Facility features.

Table 22. Server ACI Facility

Option	Description
ACI Server Definition	Create ACI server map information
Natural Extract	Extract from Natural source
COBOL Extract	Extract from a COBOL listing
PL/I Extract	Extract from a PL/I listing

Table 22. Server ACI Facility (continued)

Option	**Description**
ACI Map Display	Display ACI server map information
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps
Active Server Display	Display active ACI servers
ACI Error Create	Create ACI error processing definitions
ACI Error Display	Display ACI error processing definitions
ACI Execution Errors	Display ACI execution errors
CICS Global ACI Count	Monitor CICS global ACI counters
ACI Buffer Pools	Display ACI buffer pool information

ACI server map information

Before you can use the CALL DVS_ACI request for data, you must first define a map to the server by using the ACI Server Definition option in the ACI Facility.

Using this option, you can define an ACI server map. Users can create ACI server maps (service definitions) for the following types of servers:

- CICS servers
- Batch servers
- Stored procedures

A stored procedure is a started task that runs as a stored procedure for ACI.

The map defines and stores the definition of a remote service application. The definitions are retrieved when referenced in the second parameter of the CALL DVS_ACI request.

Defining an ACI server map

ACI server maps can be created in either of the following ways:

- Creating an ACI server map in batch
- Creating an ACI server map by using the **Server ACI Facility** panel

Creating an ACI server map in batch

Use the AVZACMP1 member (located in the *hlq*.SAVZCNTL data set) for sample JCL that you can use to create ACI server maps in batch.

Creating an ACI server map using the Server ACI Facility panel

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. From the Server ACI Facility panel, select **ACI Server Definition** and press Enter.

   The following sections guide you through creating an ACI CICS server definition and an ACI batch server definition.

Creating an ACI server definition for CICS
Create an ACI server definition for CICS.

About this task

Use the following procedure to create the ACI server definition for CICS using the Server ACI Facility.

Note: You can use the AVZACMP2 member (located in the *hlq*.SAVZCNTL data set) for sample JCL that you can use to create ACI CICS server data maps in batch.

Procedure

1. From the **Server ACI Extract** menu, select **Create ACI CICS Server Definition** and press Enter.
2. Complete the following fields:

   Note: The (R) or (O) at the end of each field indicates whether the field is Required or Optional.

   - Server Name: This value must correspond to the service information defined in the service application.
   - Server Service Class: This value must correspond to the service information defined in the service application.
   - Server Service: This value must correspond to the service information defined in the service application.

     Note:

     - The combination of the server name, server service class, and server service identifies a service.
     - If part of the name is changed while a service is active, all ACI services that are associated with the former name are treated as orphan services because an ACI service with that name no longer exists in the system. The ACI services that are associated with the former name still appear in the active ACI server maps and continue to display until they time out or until they are manually terminated.
   - Persistent Connection: Y (Yes) allows persistent connections, and N (No) allows non-persistent connections. The following differences distinguish persistent connections from non-persistent connections:
     - A persistent connection allows ongoing conversational requests and responses. The server is "assigned" to the client until the client issues an end-of-conversation (EOC) request, at which point, the ACI server program deregisters the service and terminates. When another connection requests the same ACI service, a new ACI server is started. This implies that the client and service are in conversation mode.

       It is also possible for a persistent ACI server to be reused by different client connections after the EOC request by deregistering and registering.

       Note: Reusing persistent connections improves performance and reduces overhead. For information about how to create a program to reuse persistent connections, see "Reusing persistent connections".
     - A non-persistent connection is one in which a single request is issued and a single response is received. The server is available for use by any client on a receive request. The service can be used by any incoming client connection with the Data Virtualization Manager Server.
   - Secure this Service: Y (Yes), restricts the connection to the user who has a SAF resource for the ACI service. Only a user ID with a valid resource defined for the ACI service is allowed to start and connect to that service during its life. This field defaults to N (No), which allows any user ID to start and connect to that started ACI service. The format of the resource name is:

     ```
     ACI.aci-mapname
     ```

Note: To secure a persistent ACI connection, you must edit and submit one of the following sample jobs (depending on your security type) located in your *hlq*.SAVZCNTL library to specify the map name to be used.

- AVZRAACI for RACF security
- AVZA2ACI for CA ACF2 security
- AVZTSACI for CA Top Security security

- Mirror Transaction: The name of the CICS transaction that corresponds to the EXCI mirror program DFHMIRS.
- Connection Name: The name of the CICS connection, as defined in CICS and the Data Virtualization Manager configuration member `IN00`, for DPL requests in CICS.
- Transaction Name: The user transaction to start in the CICS region.

  Note: The Mirror Transaction, Connection Name, and Transaction Name are configured by the user so the transaction can be invoked under CICS. They are previously defined in the Data Virtualization Manager configuration member.
- Unit of Work Participant (for persistent connections only): Indicates whether this transaction can process units of work. If so, the transaction must also support a persistent connection (the **Persistent Connection** field must be set to Y (Yes).
- Maximum UOW Buffer Size (for UOW participants only): The maximum buffer size that UOW transactions can accommodate for any single call (the maximum size of data that the ACI interface can send to the ACI service at any one time). The buffer size is rounded to 1000-byte increments and the maximum is 32,000. If 32,000 is specified, the ACI interface reduces that number to 31,767 bytes at execution time to comply with the maximum size the ACI service can receive at any one time.

  Note: The client can send any size data, including SQL_LONGVARCHAR and SQL_LONGVARBINARY data types, which can be greater than 31,767 bytes long. The data that is received from the client is buffered on the Data Virtualization Manager Server until a UOWLAST or UOWONLY request is received "Query syntax", at which time it is sent to the ACI service in size increments that do not exceed the maximum UOW buffer size value.
- Max Execution Time: The maximum time, in seconds, that an ACI service can run before the ACI service is set to TIMEOUT status, at which point, the client is released and receives notification that the ACI service timed out. If you do not set this value, the client non-activity timer value is used.

  For more information about the timeout values, see "Timeout values".
- Secure Server to Userid: This value defaults to N (No), which allows any user ID to reconnect to that started ACI service. To restrict the connection to the user who started the ACI service, set this value to Y (Yes). Then only the user ID that started the ACI service is allowed to reconnect to the service during its life.
- Auto Start: service Indicates to the Data Virtualization Manager server that it can start this service.
- Max. No. Allowed: The maximum number of concurrent servers of this definition type that can run at any given time.

  Note: In cases where CICS is slow or has performance problems, a client request can be submitted to multiple ACI services. To prevent a client request from being submitted to all ACI services, you can limit the number of ACI services that the client request can be submitted to "Using submisson limit checking". When the registration is requested by a CICS program running online (not started by the Data Virtualization Manager server), the Max Allowed setting does not take effect. See "Running a CICS program not started by Data Virtualization Manager server"
- Auto Terminate (for non-persistent connections only): A number 0 - 99999 to indicate the number of receives to accept before the system automatically terminates the server and the service deregisters itself. If this field is blank, the default value is 0 (zero). Complete this field only if you specified N (No) for the **Persistent Connection** field.

Note: This field limits the number of times that the server can receive requests after which it is terminated to protect storage resources.

- Client Non-Activity Timer: The non-activity timer, which has the following functions:
  - It is the amount of time that the client waits for the service to return before it times out.
  - If the max execution time value is not set, it is used for the time that an ACI service can run before timing out and releasing the client.
  - If the maximum wait for server timer value is not set, it is used for the time that a client can wait for an ACI service to be assigned before timing out.
  - For persistent services, it is also the amount of time that the service remains idle waiting for a client to converse with the service (the amount of time that is allowed for a client to interface with a service).

    Note: For persistent services only, if the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to SERVER, the Server shutdown non-activity timer value is used for all of the functions that are listed in the client non-activity timer description.

    However, the ACIPERSISTTIMEOUT would not yet apply for servers that are still in a pending registration state. Prior to registration, the Data Virtualization Manager uses the max wait for server. If that is not set, the Data Virtualization Manager uses the client non-activity timer. If the timer is not set, the DV uses a default of 15 seconds. For more information about the timeout values, see "Timeout values".
- Server Shutdown Non-Activity Timer (for non-persistent connections only): The amount of time the service can be non-active before the Data Virtualization Manager Server requests it to terminate. This field allows the less frequently used servers to stop, freeing up storage for the more frequently used servers, improving the use of available resources.

  Note: For persistent services, by default, the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to CLIENT, so this field is not used, and the client non-activity timer value is used. For more information about the timeout values, see "Timeout values".
- Maximum Wait for Server Timer: The maximum time that a client can wait for an ACI service to be assigned before the request is timed out and the client is released. If this value is not set, the client non-activity timer value is used. For more information about the timeout values, see "Timeout values".
- SDCIFEN Information: If using SDCIFEN as the program associated with the CICS transaction defined in the **Transaction Name** field to pass data to the transaction, enter the SDCIFEN information. The following information is required:
  - The name of the program to which SDCIFEN transfers control.
  - The items to be passed to the transaction using the COMMAREA.

3. Press Enter to complete the ACI server definition. If it was successfully created, the system displays the `Service is now defined` message.
4. Type the END (or press F3) to return to the **Server ACI Facility** options menu.
5. Select **Map Refresh** to refresh the data maps.

Results

To view the ACI server definitions after creating it, see "Displaying ACI server map information".

Creating an ACI batch server definition

Procedure

1. From the **Server ACI Extract** menu, select **Create ACI Batch Server Definition** and press Enter.
2. Provide the following information for the Batch ACI server definition.
   - Map Name: The name for the map.

- Server Name: This value must correspond to the service information defined in the service application.
- Server Service Class: This value must correspond to the service information defined in the service application.
- Server Service: This value must correspond to the service information defined in the service application.

  Note:

  - The combination of the server name, server service class, and server service identify a service.
  - If any part of the name is changed while a service is active, all ACI services that are associated with the former name are treated as orphan services because an ACI service with that name no longer exists in the system. The ACI services that are associated with the former name still appear in the active ACI server maps display until they time out or until they are manually terminated.
- JCL DSN: Type of JCL DSN if submitting the service as a batch job.
- Console Command: Type of `console` command if submitting the service as a started task.
- Max Allowed: The maximum number of concurrent servers of this definition type that can run at any given time.
  - In cases where CICS is slow or has performance problems, a client request can be submitted to multiple ACI services. To prevent a client request from being submitted to all ACI services, the IBM Data Virtualization Manager for z/OS Interface for ACI limits the number of ACI services that the client request can submit to as described in "Using submisson limit checking".
  - The Max Allowed setting does not take effect when the registration is requested by a CICS program running online (not started by the server). See "Running a CICS program not started by Data Virtualization Manager server".
- Persistent Connection: Y (Yes) allows persistent connections, and N (No) allows non-persistent connections. The following differences distinguish persistent connections from non-persistent connections:
  - A persistent connection allows ongoing conversational requests and responses. The server is "assigned" to the client until the client issues an end-of-conversation (EOC) request, at which point, the ACI server program unregisters the service and terminates. A new ACI server is started when another connection requests the same ACI service. This implies that the client, and service are in conversation mode.

    It is also possible for a persistent ACI server to be reused by different client connections after the EOC request by deregistering and registering.

    Note: Reusing persistent connections improves performance and reduces overhead. For information about how to create a program to reuse persistent connections, see "Reusing persistent connections" .
  - A non-persistent connection is one in which a single request is issued and a single response is received. The server is available for use by any client on a receive request. The service can be used by any incoming client connection with the Data Virtualization Manager server.
- Secure this Service: Y (Yes), restricts the connection to the user who has a SAF resource for the ACI service. Only a user ID with a valid resource defined for the ACI service is allowed to start and connect to that service during its life. This field defaults to N (No), which allows any user ID to start and connect to that started ACI service. The format of the resource name is:

  ```
  ACI.aci-mapname
  ```

  Note: To secure a persistent ACI connection, you must edit and submit one of the following sample jobs (depending on your security type) located in your *hlq*.SAVZCNTL library to specify the map name to be used.

  - AVZRAACI for RACF security
  - AVZA2ACI for CA ACF2 security

  - AVZTSACI for CA Top Security security

- Auto Start service: Indicates to the DV that it may start this service.
- Unit of Work Participant (for persistent connections only): Indicates whether this transaction can process units of work. If so, the transaction must also support a persistent connection (the **Persistent Connection** field must be set to Y (Yes).
- Maximum UOW Buffer Size (for UOW participants only): The maximum buffer size that UOW transactions can accommodate for any single call (the maximum size of data that the ACI interface can send to the ACI service at any one time). The buffer size is rounded to 1000-byte increments and the maximum is 32,000. If 32,000 is specified, the ACI interface reduces that number to 31,767 bytes at execution time to comply with the maximum size the ACI service can receive at any one time.

  Note: The client can send any size data, including SQL_LONGVARCHAR and SQL_LONGVARBINARY data types, which can be greater than 31,767 bytes long. The data that is received from the client is buffered on the Data Virtualization Manager server until a UOWLAST or UOWONLY request is received (see "Query syntax" on page 50), at which time it is sent to the ACI service in size increments that do not exceed the maximum UOW buffer size value.

- Auto Terminate (for non-persistent connections only): A number 0 - 99999 to indicate the number of receives to accept before the system automatically terminates the server and the service deregisters itself. If this field is blank, the default value is 0 (zero). Complete this field only if you specified N (No) for the **Persistent Connection** field.

  Note: This field limits the number of times that the server can receive requests after which it is terminated to protect storage resources.

- Secure Server to Userid: This value defaults to N (No), which allows any user ID to reconnect to that started ACI service. To restrict the connection to the user who started the ACI service, set this value to Y (Yes). Then only the user ID that started the ACI service is allowed to reconnect to the service during its life.
- Client Non-Activity Timer: The non-activity timer, which has the following functions:
  - It is the amount of time that the client waits for the service to return before it times out.
  - If the max execution time value is not set, it is used for the time that an ACI service can run before timing out and releasing the client.
  - If the maximum wait for server timer value is not set, it is used for the time that a client can wait for an ACI service to be assigned before timing out.
  - For persistent services, it is also the amount of time that the service remains idle waiting for a client to converse with the service (the amount of time that is allowed for a client to interface with a service).

    Note: For persistent services only, if the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to SERVER, the Server shutdown non-activity timer value is used for all of the functions that are listed in the client non-activity timer description.

    However, the ACIPERSISTTIMEOUT would not yet apply for servers that are still in a pending registration state. Before registration, the DV uses the max wait for server. If that is not set, the DV uses the client non-activity timer. If the timer is not set, the DV uses a default of 15 seconds. For more information about the timeout values, see "Timeout values" on page 48.

- Server Shutdown Non-Activity Timer (for non-persistent connections only): The amount of time the service can be non-active before the Data Virtualization Manager server requests it to terminate. This field allows the less frequently used servers to "die," freeing up storage for the more frequently used servers, improving the use of available resources.

  Note: For persistent services, by default, the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter is set to CLIENT, so this field is not used, and the client non-activity timer value is used. For more informaiton about the timeout values, see "Timeout values" on page 48.

    - Maximum Wait for Server Timer: The maximum time that a client can wait for an ACI service to be assigned before the request is timed out and the client is released. If this value is not set, the client non-activity timer value is used. For more information about the timeout values, see "Timeout values" on page 48.
3. Press Enter to complete the ACI server definition. If it was successfully created, the system displays the `Service is now defined` message.
4. Type the END command (or press F3) to return to the **Server ACI Facility** menu.
5. Select **Map Refresh** to refresh the data maps.

Results

To view the ACI server definition after you create it, see "Displaying ACI server map information".

Extracting ACI data map information

You can extract information from a Natural source, a COBOL source, or a PL/I listing. This extraction provides information about the characteristics of the program's input and output requirements.

Extracting a map from a Natural listing

You can extract a Natural listing by using either of the following methods:

- Using the AVZMFPAR member
- Using the DMF Parser

Using the AVZMFPAR member
To use this method, run the AVZMFPAR member that is located in your *hlq*.SAVZCNTL data set as a sample JCL for extracting Natural maps.

For information about the available parameters in the AVZMFPAR member, see "The batch extract member".

Using the DMF Parser

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. From the **Server ACI Facility** menu, select **Natural Extract** and press Enter.

   The **DMF Map Creation Utility** panel displays.
3. Specify the following information:
    - **Source Library Name**: The data set and member name that contains the source code for the map you want to create.
    - **Nat PGM**: The program name of your batch Natural nucleus.
    - **Load Lib**: The name of the library where the Natural nucleus resides. If you are required to concatenate multiple libraries to resolve all modules that are used during Natural execution, the library names may be used.
    - **PARM**: The Natural nucleus parameters that are required by your installation.
    - **TEMP DSN**: The name of a temporary data set that is used as a work file by this execution.
    - **Temp DSN Space**: Define this value large enough to contain the Natural object listing.
    - **Logon**: The Natural library to be logged on to.
    - **List**: The Natural object to be listed.
    - **ADARUN**: Defines the Adabas execution requirements if not linked in the Natural nucleus. After you enter the information about the panel, press Enter. The system displays a second DMF Map Creation Utility panel.
4. Provide the following information
    - **Start Field**: The field name where the map starts building.

- **End Field**: The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
- **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
- **Use Offset Zero**: If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
- **Edit Object Listing**: Edit the object listing before the data is parsed and the data map is created.
- **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.

Press Enter. The batch Natural nucleus is run to list the object you selected. Then, the ISPF editor may be invoked, depending on the Edit Object Listing selection, so that you can delete or modify information in the object listing.

5. Delete or modify information in the object listing, as appropriate. You can delete lines, fields, or information you do not want to be extracted. Leave any data elements that you want to be extracted in the editor.

   The first three lines in the ISPF editor must be deleted, even if all of the other information is required for the extract. Line 1 must be the first line as input into the extract; therefore, preceding lines must be deleted. If this is not done, the `Not Valid Source` message appears.
6. Type the END command (or press PF3) after you complete all of your edits. The data remaining in the ISPF editor is parsed and the data map is created. An `Extract Successful` message appears on the extract screen.
7. Type the END command (or press PF3) to return to the **Server ACI Facility** menu.
8. Select **Map Refresh** to add your map to the map display list.

Extracting a map from a COBOL source or COBOL/PLI listing

You can extract a map form a COBOL source, or a COBOL or PLI listing, using either of the following methods:

- Using the AVZMFPAR member
- Using the DMF parser

Using the AVZMFPAR member

Run the AVZMFPAR member that is located in your *hlq*.SAVZCNTL data set as a sample JCL for extracting COBOL and PL/I maps.

For information about the available parameters that are located in the AVZMFPAR member, "The batch extract member".

Using the DMF Parser

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. From the **Server ACI Facility** menu, select **COBOL Extract** and press Enter.

   The **DMF Map Creation Utility** panel displays.
3. Specify the following information:
   - **Source Library Name**: The data set name and member name that contain the source code for the map you want to create.
   - **Start Field**: The field name where the map starts building.
   - **End Field**: The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
   - **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.

- **Use Offset Zero**: If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
- **Convert Var to True**: Select Y (Yes) to convert VAR fields to TRUE VAR fields. TRUE VAR fields are fields that have a 2-byte length of data field that precedes the data.
- **Flatten Arrays**: Determines whether arrays are flattened. Valid values depend on the product:
  - For IBM Data Virtualization Manager for z/OS SQL, you can specify `C` (COMPATIBLE) or `Y` (YES).
  - For IBM Data Virtualization Manager for z/OS Streams, you can specify `C` (COMPATIBLE) only.
  - For IBM Data Virtualization Manager for z/OS SQL 92, you can specify `C` (COMPATIBLE), `Y` (YES), or `N` (NO).

    Note: The `C` (COMPATIBLE) value is provided for backwards compatibility with an older mapping architecture. When `C` is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
- **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.

4. Type the `END` command (or press PF3). An `Extract Successful` message appears on the extract screen.
5. Type the `END` command (or press PF3) to return to the **Server ACI Facility** menu.
6. Select **Map Refresh** to add your map to the map display list.

Displaying ACI server map information

Once the ACI servers are defined, you can view them using the ACI Map Display option. The **ACI Data Mapping Block** panel displays ACI data mapping information only. The data maps displayed in this panel represent the service, or remote application, characteristics.

About this task

To access the **ACI Data Mapping Block** panel:

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **ACI Map Display** from the **Server ACI Facility** menu. Press Enter.

   The system displays the **ACI Data Mapping Block** panel.

   Note: Some active ACI server information can be returned in a result set using a simple query with the IBM Data Virtualization Manager for z/OS driver. For more information, see "Using a query" on page 45.

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	**Description**
`P`	Prints map.
`S`	Shows map.
`D`	Disables map.
`E`	Enables map.
`M`	Modifies/displays map.

The `M` command can be used to display or modify an ACI server definition. If no fields are changed on the panel that is displayed, the map is not saved. If any field is changed, the map is saved and a refresh is required to make the changes active. Changes cannot be viewed until a refresh is done.

Column names

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description
STRUCTURE NAME	The ACI server map name.
STATUS	The active status of the ACI server.
MAX. NO. SERVERS	The maximum number of services that can be running concurrently.
ACTIVE SERVERS	The number of services that are currently running.
HIGH WATER SERVER USAGE	High water marks concerning service usage.
REGISTER COUNT	The number of times a service registers. Incremented when a REGISTER completes.
DEREG COUNT	The number of times a service deregisters. Incremented when a DEREGISTER completes.
SEND COUNT	The number of buffers a service has sent to Data Virtualization Manager server. Incremented when a SEND completes.
RECEIVE COUNT	The number of requests a service has received from a client. Incremented when a RECEIVE or RCV ON SND completes. **Note:** RECV READY is not counted in this count because a RECV READY means that the program is waiting for a client. As soon as the client connects, a RECEIVE occurs and this is when the count is incremented.
TIMEOUT COUNT	The number of times a client has timed out waiting for a server (see “Timeout values” on page 48). Incremented when a client request (CALL DVS_ACI) times out while waiting for an ACI server to be available.
ABEND COUNT	The number of times a server has terminated abnormally. Incremented when an ACI server service abends.
WAIT COUNT	The number of times a client has been waiting for an available server. Incremented each time a CALL DVS_ACI request waits for an ACI service (for example, a WAITING FOR THE SERVER occurrence).

Note: The counts that are displayed on this panel are reset when the server is restarted.

If an error definition isdefined see “Displaying CICS global ACI counters” on page 46, the columns that are described in the following table also contain information.

Column name	Description
SUSPEND COUNT	The number of times a server has been suspended because of an error.
LAST SUSPENDED DATE TIME	The last time a server was suspended.
SUSPEND ERROR	The error that caused the server to abend.
SUSP_SEC REMAINING TOTAL	The time (in seconds) until the server resumes.
TOTAL ERRORS	The total number of errors that are received by the server.
INACTIVE TIMEOUTS	The number of inactive timeouts.
MODIFICATION DATE TIME	The date and time the map was modified.
USERID	The user ID of the map creator.

Displaying all map information
The **Data Mapping Block** panel displays all data maps that are defined to this Data Virtualization Manager server.

About this task

To display map information:

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **Map Display** from the **Server ACI Facility** menu. Press Enter.

   The system displays the **Data Mapping Block** panel. Several panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
D	Disables the map so it is unavailable for use.
E	Enables the map for use.
K	Deletes a map, making it unavailable for use.
P	Prints the associated control block for the selected row.
S	Displays the associated control block for the selected row.
X	Displays the map elements for the selected row.

Column names

The following table provides a description and sort name (if available) for each column name on the ISPF panels.

Column name	Description	Sort name
STRUCTURE NAME	The data map name.	NAME
TYPE	The type of data map.	TYPE
STATUS	The status of the service: • Enabled • Disabled • Deleted	STATUS
MR	The MapReduce status. Y indicates enabled and N indicates disabled.	MR
LANGUAGE	The language type from which this map was generated.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The creation date and time of this map.	DATE
USERID	The user ID of the map creator.	USERID
CREATION DATASET	The data set from which the map was extracted.	DATASET

Copying ACI maps

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI**.
2. Then, select **Map Copy** from the **Server ACI Facility** panel. Press Enter.

   The system displays the Move/Copy Utility panel.
3. Type one of the following commands in the **Option** field:
   - C to copy
   - CP to copy and print
   - M to move
   - MP to move and print
4. In the From ISPF Library fields, provide the information for the data set, including values for the Project, Group, and Type information. If the data set is partitioned, type a member name in the **Member** field:
   - To move, copy, or promote a single member, type the member name.
   - To move, copy, or promote all members, type * (asterisk).
   - To request a member selection list, leave the member name blank or specify a pattern.

   Alternatively, for any other partitioned or sequential data sets, you can specify the From Other Partitioned or **Sequential Data Set** field. Type the data set name and volume serial number. Press Enter.

   Note: If you forget to enter a password for a data set that requires one, or if you enter the password incorrectly, the system prompts you in standard TSO (line) mode. On TSO/TCAM systems, you may need to press the CLEAR key before responding to the password prompt. If you enter the password incorrectly or encounter any other problems, you may be prompted again to enter the password until you reach a system limit of attempts.

Displaying active ACI server information
You can display active ACI server information in either of the following ways:

- "Using the active server display" on page 44
- "Using a query" on page 45

The information that displays represents active services or remote applications that are running in the system. These services are registered to this Data Virtualization Manager server instance and are assigned a server ID. The server name is the same as that defined in the service definition.

Using the active server display

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **Active Server Display** from the **Server ACI Facility** panel display and press Enter. The system displays the **ACI Servers** panel.

   Two panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

   Note: Some active ACI server information can be returned in a result set by using a query. For more information, see "Using a query" on page 45.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
P	Prints the map.
S	Shows the map.
K	Terminates the map.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description
SERVER ID	The server ID.
SERVER NAME	The name of the server as defined in the service definition.

Column name	Description
STAT	The status of the service: • 0: Waiting for work from a client. • 1: Busy or assigned conversationally to a client. • 2: Registered but not assigned. • 3: Deregistered but not released. • 5: Waiting for the program to terminate or reset. • 6: EOC issued waiting for service to process command. • 7: Start issued waiting for service to register. **Note:** Status 4 is not used.
LAST ACTIVE	This value depends on the status of the service: For status 0, this is the number of seconds that the service has been idle. For other status values, this is the number of seconds that the service has been in use.
CONN ID	The CICS connection name or load balancing name
TRAN ID	The CICS transaction name running in the CICS region for this service.
TASK ID	The CICS task ID running in the CICS region for this service.
MAXIMUM LAST ACTIVE	The high-water mark for the LAST ACTIVE count. **Note:** The MAXIMUM LAST ACTIVE column displays on the next panel. Use the RIGHT scroll commands (or PF11 key) to scroll to the right.

Using a query

Using the driver, your application can return active ACI server information in a result set by using a query.

About this task

The syntax of the query is:

CALL DVS_INFO('ACTIVEACISERVERS','*optional-filters*')

where:

- ACTIVEACISERVERS (Required) causes the query to return a result set with all active ACI servers listed.
- *optional-filters* (Optional) Specifies a filter for the query. Valid filters are:
  - NAME (*server-name*): Obtains results for the server name specified. For example:

    ```
    CALL DVS_INFO('ACTIVEACISERVERS','NAME(SDCIFEN)')
    ```

  - CONNECTION (*connection-name*): Obtains results for the CICS connection name or load balancing name specified. For example:

    ```
    CALL DVS_INFO('ACTIVEACISERVERS','CONNECTION(EXCS)')
    ```

  - PERSIST (YES|NO|ALL): Obtains results for servers with the persistent status specified:

- YES selects persistent servers.
- NO selects non-persistent servers.
- ALL (Default) selects all servers (persistent and non-persistent).

  For example:

  ```
  CALL DVS_INFO('ACTIVEACISERVERS','PERSIST(ALL)')
  ```

Displaying CICS global ACI counters

With the CICS Global ACI Count option, you can display the current values of MAXTASKS and the number of ACI services that are running for each CICS that is running ACI services. You can also display the last service to update the counter. If the counter is determined to be inaccurate, the counter can be updated by typing over the counter-value.

About this task

To display CICS Global ACI Counters:

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **ACI** and press Enter.
2. Select **CICS Global Count** from the **Server ACI Facility** panel and press Enter. The system displays the **Global ACI Counters Display** panel.

   Use the LEFT and RIGHT scroll commands (or PF keys) to shift between the two panels that display the global ACI counters.

   Note: The maximum number of ACI services started is managed globally among all Data Virtualization Manager servers. This limits the number of ACI services even when the ACI configuration would otherwise allow more services to start. Refer to the DEFINE CONNECTION statement, MAXTCUSHION parameter. This defines a value that is used to further limit the number of ACI services that can be started. The MAXTCUSHION value is subtracted from the MAXTASKS value found in CICS, and used to reserve some tasks for non-ACI work in CICS.

   Use the X line command to view all of the ACI Servers for a CICS APPLID.

Converting program data types to ODBC

COBOL conversions

COBOL conversions describe how COBOL data types are converted to ODBC data types.

Table 23. Conversions of COBOL data types to ODBC data types

COBOL	ODBC
Alphanumeric	SQL_CHAR
Floating Point	(If 4 bytes) SQL_FLOAT (If 8 bytes) SQL_DOUBLE
Integer	(If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
Numeric	SQL_NUMERIC
Packed	SQL_DECIMAL

Natural conversions
Natural conversions describe how Natural data types are converted to ODBC data types.

Natural	**ODBC**
A-Alphanumeric	SQL_CHAR
B-Binary	(If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
C-Attribute Control	N/A
D-Date	*SQL_DECIMAL
F-Floating Point	(If 4 bytes) SQL_FLOAT (If 8 bytes) SQL_DOUBLE
I-Integer	(If 1 byte) SQL_BINARY (If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
L-Logical	SQL_BINARY
N-Numeric	SQL_NUMERIC
P-Packed	SQL_DECIMAL
T-Time	*SQL_DECIMAL

Note: Although the IBM Data Virtualization Manager for z/OS Interface for Adabas supports the conversion of ODBC date and time to the Natural date and time format, the IBM Data Virtualization Manager for z/OS Interface for Natural only allows the passing of the internal format for date and time (P6 and P12, respectively).

Reusing persistent connections
Persistent connections can be reused so that the ACI service can be used by different client connections. The number of times that a service can be reused is controlled by the application, not by the IBM Data Virtualization Manager for z/OS Interface for ACI.

Once the client issues the end-of-conversation (EOC) request, the ACI service and its CICS task normally become unavailable. For the persistent service to be reused, the program must perform one of the following actions:

- DEREGISTER and REGISTER again. Then, go into the RECV READY state.
- Enter the RECV READY state by using `CONV-ID = 'NEW'`. In this case, the IBM Data Virtualization Manager for z/OS Interface for ACI implicitly issues a DEREGISTER/REGISTER on the client's behalf.

  Note: Although the ACI service is reused, the server ID of the service is changed each time because of the DEREGISTER/REGISTER calls.

Using submisson limit checking
The ACI interface limits the number of ACI services that can be submitted for the client request. When a request to start an ACI server is received from a client, the ACI interface attempts to start the ACI server or waits for a short interval depending on the following criteria:

- The active ACI server queue is searched for an available ACI server that matches the ACI service definition, as configured in the data map. If found, that server is assigned to this request and processing continues.

  For more information about creating an ACI service definition, see "Defining an ACI server map" on page 32
- If either of the following situations occur, the request waits for a short interval:

  – The number of active ACI servers is greater than the Max Allowed setting of the ACI service definition, which specifies the maximum number of concurrent servers allowed.
  – The number of start attempts for this request is greater than the maximum allowed (five).

Note: The first criterion that is met determines the action taken.

If no start attempts have been made for this request, a start attempt is made. The current registration count for this ACI service definition is saved.

If the wait interval is less than a second, the request waits for another short interval.

If a start attempt is made, but the current registration count for this ACI service definition has changed, another requestor may have obtained control of the ACI server. A new ACI server is started and the current registration count for this ACI service definition is saved.

For other situations, the request waits for a short interval as defined. The interval time that a request waits starts at 0.25 seconds and doubles for each waiting interval until it reaches a maximum of 5 seconds.

Note: The `WAITING FOR SERVER` message does not appear until after the interval reaches five seconds.

The maximum amount of time that a request waits for an ACI server to be assigned is defined by the Maximum Wait For Server Timer value in the ACI service definition; otherwise, the Client Non-Activity Timer value is used.

When the maximum amount of time is reached for the client to wait for an available server, the request terminates with an error, depending on whether any servers are active for this ACI service definition:

- If at least one server is active for this ACI service definition:

```
DVS_ACI ERROR HAS OCCURRED RC -1062; TIMEOUT EXCEEDED, ALL SERVERS ARE BUSY
```

- If no servers are active for this ACI service definition:

```
DVS_ACI ERROR HAS OCCURRED RC -1081; NO ACI SERVICE AVAILABLE / CANNOT
START ACI SERVICE - CHECK FOR SERVER FAILURE
```

Timeout values
Timeout values describe the amount of time to wait before timing out in the ACI server definition, how each timeout value is set, and the resulting error message that is returned to the requesting application.

Table 24. ACI timeout values

Event Description	Method of control	Client error code returned

The timeout for a client waiting for an available server (the amount of time that the client waits for a service connection).	• Maximum Wait for Server Timer
	• If the Maximum Wait for Server Timer value is not specified, the Client Non-Activity Timer value is used.
		If no server is active:
		`DVS_ACI ERROR HAS OCCURRED RC -1081; NO ACI SERVICE AVAILABLE / CANNOT START ACI SERVICE - CHECK FOR SERVER FAILURE`
		If a server is active, but unavailable:
		`DVS_ACI ERROR HAS OCCURRED RC -1062; TIMEOUT EXCEEDED, ALL SERVERS ARE BUSY`

Table 24. ACI timeout values (continued)

Event Description	**Method of control**	**Client error code returned**
The timeout value for a client waiting for a server to return (the time that is allowed for a service to complete a unit of work before the result is sent to the client).	Client Non-Activity Timer	`DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT`
The maximum server execution time (the time that is allowed for a server to run).	• Max Execution Time • If the Max Execution Time is not specified, the Client Non-Activity Timer is used.	`DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT`
The timeout value for an idle server waiting for a client to make a request (the amount of time the service can be non-active before Data Virtualization Manager server requests the service to terminate).	• Non-Persistent Connections: Controlled by the Server Shutdown Non-Activity Timer. • Persistent Connections: Controlled by the Client Non-Activity Timer.	

Note: For persistent connections, the method of controlling timeout values depends on the value of the ACIPERSISTTIMEOUT (ACI PERSISTENT SERVER TIMEOUT) parameter:

- If the parameter `ACIPERSISTTIMEOUT = CLIENT` (default) is defined, the client non-activity timer value is used.
- If the parameter `ACIPERSISTTIMEOUT = SERVER` is defined, the server shutdown non-activity timer value is used for all of the client non-activity timer functions.

Handling interrupted connections
Interrupted connections affect the following items:

- ACI service status
- Client error codes

ACI service status

When the ACI service is busy in status 1, but the connection is interrupted while issuing a CALL DVS_ACI, the IBM Data Virtualization Manager for z/OS Interface for ACI ensures that the situation is handled appropriately by marking the connections as timed out. This allows the server to clean up and deregister. The server is placed in status 5, which indicates that it is waiting for the application to terminate or to reset.

The ACI service remains in status 5 until the application responds by using a SEND/RECEIVE command. Once the SEND/RECEIVE is received, the application receives a TIMEOUT error code (`#ETBCB-ERROR-CODE = TIMEOUT`). The application then issues a DEREGISTER, and the ACI service is cleaned up.

Client error codes

The client receives an appropriate error code, depending on which of the following occurrences caused the interrupted connection:

- Connection Timing Out: If the application reaches the timeout setting while waiting for a server to return or waiting for server execution (see "Creating an ACI server definition for CICS" on page 33 for description) while issuing a CALL DVS_ACI, the application receives the following message:

```
DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT
```

Note: In the case of persistent services, subsequent calls to this service get the following message:

```
DVS_ACI ERROR HAS OCCURRED RC -1065; SERVER HAS NOT RESPONDED, TIMEOUT
```

The service that is assigned to the client must be terminated so the client can restart another persistent service and start a new conversation. Once the service is terminated, any subsequent calls to this service receive the following message:

```
DVS_ACI ERROR HAS OCCURRED RC -1071; CONVERSATION HAS NOT BEEN
ESTABLISHED OR IS TIMED OUT BY SERVICE
```

The client must start a new conversation.

- Terminated Connection: If the connection was terminated, the client receives the following message:

  ```
  Host Communication Failed
  ```

  Connections can be terminated by the following methods:

  - Data Virtualization Manager server FAILxxxxxTIME parameter, which terminates the connection if the connection exceeds the value specified.
  - `Kill` line command of the Remote User program (accessed from the Data Virtualization Manager server - Primary Option Menu).

Running a CICS program not started by Data Virtualization Manager server
CICS programs that are not started by Data Virtualization Manager server can register with Data Virtualization Manager server by using the SDBRTX table.

Note: The Max Allowed setting in the ACI service map does not take effect when registration is requested by a program that is not started by Data Virtualization Manager server because this setting limits the number of CICS transactions (the number of programs) that can be started by Data Virtualization Manager server. The MAX NO SERVERS and MAX ACTIVE SERVERS counts in the ACI server maps display do not apply for this type of registration scenario.

When registration is requested by a CICS program that is not started by Data Virtualization Manager server, registration process performs the following actions:

- Determines the Data Virtualization Manager server subsystem. The SDBRTX table is checked to see if an entry with the transaction name matches the transaction name under which the program is running:
  - If a match is found, the registration goes to the Data Virtualization Manager server subsystem specified in this entry.
  - If no match is found, the subsystem name on the default entry is used.
- Determines the ACI service. The ACI service is determined in the following ways:
  - If ACIDEFAULTCONNNAME ((ACI DEFAULT CONNECTION NAME) is not set, the IBM Data Virtualization Manager for z/OS Interface for ACI bypasses connection name checking. The first ACI service with a triple name that matches the triple name that is specified by the program is used for the registration process.
  - If ACIDEFAULTCONNNAME is set, the IBM Data Virtualization Manager for z/OS Interface for ACI enables connection name checking, which means that an ACI service is used for the registration process only if the triple name matches the triple name that is specified by the program and the connection name matches the value of ACIDEFAULTCONNNAME. If no match is found, the registration request receives an ACI error code of `01000100`.

Query syntax
The syntax of a query is:

CALL DVS_ACI('*function*','*datamaps*','data1',...,'dataN')

where

`function` is `SEND`, `SOC`, `EOC`, `UOWFIRST`, `UOWMIDDLE`, `UOWLAST`, or `UOWONLY`:

- SEND: The data strings are sent to the server defined in the server data map.

  Note: The SEND function implies that you receive information in return.
- SOC: Start of conversation. This function is required for persistent servers. It is used to obtain an existing service or to start a server and lock a server from use by the client.
- EOC: End of conversation. This function is required for persistent service. It is used to notify the service that it is no longer registered to a client.
- UOWFIRST: Indicates that this message is the first part of a unit of work (UOW). The messages are accumulated on the Data Virtualization Manager server until a request indicates a function of UOWLAST is received.
- UOWMIDDLE: Indicates that this message is not the first part or the last part of a UOW. The messages are accumulated on the Data Virtualization Manager server until a request indicates a function of UOWLAST is received.
- UOWLAST: Indicates that this message is the last part of a UOW. When this is received, the Data Virtualization Manager server processes the entire UOW, sending the messages to the ACI service in the size increments it desires.
- UOWONLY: Indicates that this is a one-message UOW. When this is received, the Data Virtualization Manager server processes the UOW, sending the messages to the ACI service in the size increments it desires.

  Note: For a UOW, the size of the message segments that are sent by the client are not dependent on the size that the ACI service can accept. Any size segment can be sent by the client by using the SQL_LONGVARCHAR and SQL_LONGVARBINARY data types.

datamaps are the data maps (up to three data map names):

- (Required) Specifies the map that defines the server to which the request is being assigned. The map name is required.
- (Optional) Client map in (CMI). Defines the client input (data1-dataN) presented to the server. The CMI represents the data format expected by the service.

If CMI is coded, the data parameters data1-dataN are validated as described in “Data validation”.

Note:

- For special considerations on passing numeric data with CMI, see “Passing numeric data”.
- CMI is not supported for UOW calls. If a CMI is specified for a UOW call, the following message is generated:

  ```
  DVS_ACI ERROR HAS OCCURRED RC -1086; INPUT DATA MAP NOT ALLOWED FOR UNIT
  OF WORK TRANSACTIONS
  ```
- (Optional) Server map output (SMO). Describes the data as presented by the server. If SMO is coded, the data buffer output from the source is presented as a result set described by the SMO data map.

  If the SMO is not specified, the Data Virtualization Manager server cannot determine the maximum size of the row in the result set until the first SEND call of each CALL DVS_ACI invocation is made. Once the first SEND call is issued, the Data Virtualization Manager server uses the length of the first SEND call to establish the maximum size of the row in the result set.

  Note the following guidelines:

  – If an optional CMI and SMO are specified, separate them by commas.

  – If a CMI is omitted and an SMO is specified, use a comma as a placeholder for the CMI.

  – You can run a simple CALL statement to return metadata for the CMI or SMO.

data1-dataN describes the data input to the server. If more than one data area is coded, a CMI is required.

CMI considerations

When passing an ACI input map (CMI), remember the following considerations:

- Data validation
- Passing numeric data

Data validation

If CMI is coded, the data parameters data1-dataN are validated in the following ways:

- If only one data parameter is given, the CMI is used for validation of data types only; that is, numeric fields are numeric.
- If more than one data parameter is given, the CMI is used to validate and buffer the data components as input to the server.

Passing numeric data

The following considerations exist for passing numeric data with a CMI:

- **Packed Decimal Fields**. If a field is defined as Packed Decimal in the ACI input map, the following guidelines apply:
  - If the value passed has a scale that is too long, the size of the scale in the ACI input map is used. The IBM Data Virtualization Manager for z/OS Interface for ACI allows the value if the adjusted precision is less than or equal to the precision in the ACI input map, and the scale is truncated.
  - Although the IBM Data Virtualization Manager for z/OS Interface for ACI allows you to pass a string to a Packed Decimal field, it does not allow the decimal point to be specified in the string (the decimal is based on what is defined in the ACI input map). Also, if the length of the string exceeds the precision of the field, the leading digits in the string are truncated. For these reasons, it is not recommended to pass a string value to a packed field.
- **SmallInt Fields**. The IBM Data Virtualization Manager for z/OS Interface for ACI allows a value that is passed as an integer to a field defined in the ACI input map as SmallInt. Ensure that the value is less than or equal to 32767. Otherwise, the data is truncated.

Using a CALL statement to obtain map metadata

The IBM Data Virtualization Manager for z/OS Interface for ACI allows users to view metadata information for CMI and SMO maps on the client with a simple CALL statement. This allows a user to pass all the input parameters with the correct data types as required by the CMI or SMO without having to go into the server ISPF panels to locate this information.

The format of the CALL statement is:

CALL DVS_MAP('DESCRIBE','*mapname*')

where *mapname* is the name of the map.

The CALL statement returns a result set with a single column named FORMAT. This column contains details on the fields of the map. The following table describes the FORMAT column types and their SQL equivalents.

Table 25. FORMAT column types and the SQL equivalent

FORMAT types	SQL types
CHARACTER	SQL_CHARACTER
NUMERIC	SQL_NUMERIC
DECIMAL	SQL_DECIMAL
INTEGER	SQL_INTEGER
SMALLINT	SQL_SMALLINT
FLOAT	SQL_FLOAT

Table 25. FORMAT column types and the SQL equivalent (continued)

FORMAT types	SQL types
DOUBLE	SQL_DOUBLE
DATE	SQL_DATE
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP
VARCHAR	SQL_VARCHAR
LONGVARCHAR	SQL_LONGVARCHAR
BINARY	SQL_BINARY
VARBINARY	SQL_VARBINARY
LONGVARBINARY	SQL_LONGVARBINARY
UNICODE	SQL_UNICODE
UNICODE_VARCHAR	SQL_UNICODE_VARCHAR
UNICODE_LONGVARCHAR	SQL_UNICODE_LONGVARCHAR

IBM Data Virtualization Manager for z/OS Interface for Adabas

The IBM Data Virtualization Manager for z/OS Interface for Adabas allows ODBC, JDBC, and Web clients to access Adabas data in a relational model by using simple SQL-based queries. This interface can be used with traditional client/server applications, desktop productivity tools that use ODBC, and 2-tier and 3-tier Web implementations. Using the IBM Data Virtualization Manager for z/OS Interface for Adabas, any ODBC- or JDBC-enabled application can use standard ODBC or JDBC facilities to make SQL requests directly to Adabas. The result is a relational result set, with no host programming required.

The Adabas Interface Facilities option on the Data Virtualization Manager server - Primary Option Menu provides access to the Server Adabas Data Mapping Facility features.

Table 26. Server Adabas Data Mapping Facility

Option	Description
Map Defaults	Set map options
Map Create	Create maps
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps

Creating Adabas virtual tables using the Data Mapping Facility in batch

To extract and import Adabas data, use the sample JCL in the AVZMFPAR member.

Member AVZMFPAR, which is in the *hlq*.SAVZCNTL data set, contains sample JCL for extracting Adabas virtual tables.

For information about the available parameters in the AVZMFPAR member, see "The batch extract member".

IBM Data Virtualization Manager for z/OS Interface for CICS/TS

The CICS/TS interface offers quick and easy access to CICS/TS, making CICS data directly available to non-mainframe users.

The Data Virtualization Interface for CICS/TS provides a tool for making quick and easy CICS/TS queries. IBM Data Virtualization Manager for z/OS, in conjunction with this interface, allows existing and new CICS programs to be rapidly integrated into client/server applications with minimum modification to the code.

The Data Virtualization Manager server component of IBM Data Virtualization Manager for z/OS connects to CICS by using the IBM EXCI (External CICS Interface). In IBM Data Virtualization Manager for z/OS, a CICS program behaves like a stored procedure. Applications can call the procedure by using industry-standard mechanisms, passing in the input data as part of the call. The CICS program returns data by using the COMMAREA to the Data Virtualization Manager server. The Data Virtualization Manager server, through the Data Mapping Facility, formats the display of the returned results.

The CICS Interface Facilities option on the IBM Data Virtualization Manager for z/OS - Primary Option Menu provides access to the Server CICS Control Facility features.

Table 27. Server CICS Control Facility

Option	**Description**
CICS Connections	Monitor and Control CICS Connections
CICS Sessions	Monitor and Control CICS Sessions
CICS Data Mapping	Create CICS Map Information
CICO	CICS Global Information Control Block
EXCI	EXCI Global Information Control Block
CICS Load Balancing	Monitor CICS Load Balancing
CICS Global ACI Cnt	Monitor CICS Global ACI Counters

Choosing a connectivity method

Before you use the IBM Data Virtualization Manager for z/OS Interface for CICS/TS, decide which of the following connectivity methods best fits your needs:

- Stored Procedures (metadata support)
  - Custom RPC, a sample of which is DVSCIBSC
  - ODBC CICS RPC, a sample of which is SDCOCIRP

Stored procedures

A stored procedure contains the metadata for input and output fields, as well as other required parameters for accessing CICS transactions. It stores this information in the Data Mapping Facility.

To use a stored procedure to perform a CICS/TS query, see "Creating a stored procedure by using the CICS Control facility" on page 55.

Remote Procedure Calls (RPCs)

The RPC types provide the following benefits:

- Custom RPC, DVSCIBSC (sample) provides users a way to invoke multiple CICS programs and return a customized result set to the client in only one network round trip.
- ODBC CICS RPC, SDCOCIRP (sample) provides users a way to eliminate the restriction of a 32 KB COMMAREA.

Custom RPC, DVSCIBSC (sample)

The sample Custom RPC, DVSCIBSC, can be run with a CALL statement from any ODBC- or JDBC-compliant application, such as Visual Basic, PowerBuilder, and MS-Access. Because the applications are developed by using APIs provided on the host, they can return an ODBC or a JDBC result set to the application. Using custom RPCs, the RPCs run and end similar to the way a normal DB2 query runs and ends. The results of the RPC are accessed in the same manner as the results of a DB2 query.

The source for the sample RPC, DVSCIBSC, is on the z/OS host in the *hlq*.SAVZSAMP data set. Compiled copies of these samples are also provided in the *hlq*.RPCLIB data set.

Note: This call is under the restriction of the 32 KB limitation of the COMMAREA.

To use the custom RPC to perform a CICS/TS query, see "Using the custom RPC, DVSCIBSC (sample)" on page 59.

ODBC CICS RPC, SDCOCIRP (sample)

The ODBC CICS RPC, SDCOCIRP, is a remote procedure call that runs in the CICS address space and uses IBM Data Virtualization Manager for z/OS APIs to avoid the restrictions of the 32 KB COMMAREA. ODBC CICS RPCs enable non-CICS applications that run non-z/OS operating systems to access enterprise data and applications that are managed by CICS.

Using this connection option, the driver can make a single network roundtrip by using multiple programs, and return a single customized result set of the combined data. For example, if you are running five programs, and you want specific columns of data in each program, the ODBC CICS RPC delivers only one set of with the specified data. You can also return zero or multiple result sets from an ODBC CICS RPC.

In addition, this connection option allows you to connect to different CICS regions from a single Data Virtualization Manager server.

To use the ODBC CICS RPC to perform a CICS/TS query, see "Using the ODBC CICS RPC, SDCOCIRP (sample)" on page 59.

Creating a stored procedure by using the CICS Control facility

You can use the CICS Control facility to create a map extract by using COBOL or PL/I and generate a stored procedure.

Extracting a map from a COBOL and/or PLI listing

You can extract a COBOL and/or PLI listing using either of the following methods:

- Using the AVZMFPAR member
- Using the DMF parser

Using the AVZMFPAR member

To use this method, run the AVZMFPAR member that is located in your *hlq*.SAVZCNTL data set as a sample JCL for extracting CICS maps.

For information about the CICS parameters that are located in the AVZMFPAR member, see "The batch extract member".

Using the DMF parser

About this task

Use the DMF parser to extract a map from COBOL. The procedure is similar for the PLI extract. For information about extracting a VSAM map, see "Creating data maps for VSAM file access".

Procedure

1. From the Primary Option Menu, select **CICS** and press Enter.
2. Select **CICS Data Mapping** from the CICS Control Facility Display and press Enter.
3. Make sure the Parser Version is set to `N` (New).
4. Return to the **CICS Data Mapping** menu and select **Extract Cobol** and press Enter.
5. Provide the following information:

- **Source Library Name**: The data set name and member name that contain the source code for the map you want to create.
- **Start Field**: The field name where the map starts building.
- **End Field**: The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
- **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
- **Use Offset Zero**: If the Start Field is not an '01' level, start the offset at zero. Otherwise, the offset starts at the offset of the field in the structure.
- **Flatten Arrays**: Determines whether arrays are flattened. Valid values depend on the product:
  - For IBM Data Virtualization Manager for z/OS SQL, you can specify `C` (COMPATIBLE) or `Y` (YES).
  - For IBM Data Virtualization Manager for z/OS Stream, you can specify `C` (COMPATIBLE) only.
  - For IBM Data Virtualization Manager for z/OS SQL 92 Engine, you can specify `C` (COMPATIBLE), `Y` (YES), or `N` (NO).

    Note: The `C` (COMPATIBLE) value is provided for backwards compatibility with an older mapping architecture. When `C` is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
- **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.

6. Press Enter. If the extract completes with no errors, the `Extract Successful` message appears in the upper right corner of the panel. Both the map library and the Data Virtualization Manager server contain the mapped structure definition.
7. Return to the **CICS Data Mapping** panel and select **Map Refresh** to add your map to the map display list.

   Note: The output from the extract is a virtual table definition that is placed in the named map library. The virtual table library member name is the name that is associated for this virtual table by Data Virtualization Manager server.

   After virtual table generation, you must disable virtual table fields that are not used by your application by using the Map Display option of the Data Virtualization Manager server CICS Data Mapping panel.

Generating a stored procedure by using data map(s)

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **Generate a Stored Procedure from Maps** and press Enter.
2. Provide the following information:
   - **Map Data Set Library/Other Map Data Set Name**: The DSNAME of the map data set. This is the location of the input and output maps that are used to create the stored procedure and where the stored procedure map is created.
   - **Map Names**: The input and the output maps that were created. In this example, the same map is used for input and output.
   - **Procedure Name**: The name of the CICS procedure to create. This is written to the DMF.
   - **Interface Type**: Type C (CICS).
   - **CICS Transaction ID**: A mirror transaction name. If you are unsure of the mirror transaction name, type EXCI, the default value. This name is used to communicate with the CICS address space.

     Note: If you specify a null value for the CICS Transaction ID, IBM Data Virtualization Manager for z/OS issues the DPL request with a NULL pointer for the transaction ID address to allow for CICS transaction dynamic routing. In addition, this implementation allows your application to specify NULL in the client setting Transaction Name (TRNA). If this keyword is set, any CICSEX request that is issued by the application results in a DPL request with a NULL pointer for the transaction ID address that is passed to CICS.

- **CICS Transaction ID2**: The TRANID2 associated with CICS transaction, if applicable.
- **CICS Connection Name** or **CICS Load Balancing Group Name**: The CICS connection name or CICS load balancing group name. This name is used for the DPL request at run time.

  Note: If you do not specify the connection name of load balancing group name when creating the map, the name is picked up from the EXCICONNECTIONNAME parameter (this is the EXCI DEFAULT CONNECTION NAME parameter in the IBM Data Virtualization Manager for z/OS started task parameter PRODCICS group).
- **CICS Program Name**: The name of the CICS program to be invoked. This program must have a valid PPT entry that is defined to CICS.
- **Dynamic Connection** and **TRANID Support**: An optional map that is used to dynamically change the default connection name or load balancing name and TRANID at execution.

  By default, the stored procedure that is defined uses the CICS connection name or load balancing name and TRANID defined for the stored procedure. This definition can be changed at client connection time by using the ODBC/JDBC client CNNA and TRNA connection keywords. Using the client CNNA and TRNA keywords to override the defaults requires that a new connection be established. To change these defaults at execution time, you can add input parameters either at the beginning (PREFIX) or end (SUFFIX) of the stored procedure input parameter list. By specifying PREFIX or SUFFIX, input parameters CICS_CONNECTION and CICS_TRANSACTION are automatically added to the stored procedures input map definition (the original input map that is supplied is not modified.). These are optional parameters and if set to '" (a 0-length string), existing defaults are used. If set to another value at execution time, the new value is used to run the CICS stored procedure.

3. Return to the **CICS Data Mapping** panel and select **Map Refresh** to add your map to the Map Display list.

   The pseudo-stored procedure name is returned by the ODBC function call SQLProcedures when directed at the Data Virtualization Manager server address space. The contained metadata is returned by SQLProcedureColumns so that a SQLExecute can be used to invoke the procedure. IBM Data Virtualization Manager for z/OS maps all application parameters as mapped by the previous extract and passes them to the CICS program. All output is passed according to the previously mapped output map.

   The procedure generation process uses the value that is specified in the IBM Data Virtualization Manager for z/OS started task parameter CICSPROCOWNER. The default for the CICS stored procedure owner CICSPROCOWNER is CICSEX. CICSPROCOWNER can be found in the IBM Data Virtualization Manager for z/OS started task parameter group PRODCICS.
4. Return to the **CICS Data Mapping** panel and select **Map Display** to view the map display list and press Enter.

   This example shows the first of two panels for displaying existing data maps. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
P	Prints the associated control block for the selected row.
S	Displays the associated control block for the selected row.
D	Causes map to be unavailable for use.

Line commands	Description
E	Enables the map for use.
K	Deletes a map, also making it unavailable for use.
X	Displays map elements for the selected row.

Type the command to the left of the line and press Enter.

Column names
The following table describes each column name on the ISPF panels and provides a sort name.

Column name	Description	Sort name
STRUCTURE NAME	The member names in the map data set.	NAME
TYPE	TYPE • ADABAS • Input • Output • Screen • LPTBL • Header • USER	TYPE
STATUS	The status of this map (Enabled, Disabled, or Deleted).	STATUS
LANGUAGE	The language of the extracted map. This value is determined at the time of the extract.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The date and time the map was last modified.	DATE
USERID	The user ID of the map creator. Used only for informational purposes.	USERID
CREATION DATASET	The data set from which the map was extracted.	DATASET

Query syntax: accessing a CICS program by using a stored procedure
The syntax of a query is:

Call *owner.procedure-name*(*parm1*,*parm2*,...)

where:

- *owner* is the name that is defined in the Started Task parameter CICSPROCOWNER. In the following example, CICSEX is the owner.
- *procedure-name* is the stored procedure name that invokes the transaction. In the following example, the stored procedure is TESTPROC.
- *parm1,parm2* defines the parameters to be passed to the CICS transaction. These parameters are defined by using the map extraction procedure in the IBM Data Virtualization Manager for z/OS ISPF

interface. These parameter descriptions are passed to the program by using the SQLProcedureColumns function call.

For this example, the actual call would be:

```
Call CICSEX.TESTPROC(parm1,parm2,...)
```

Using the custom RPC, DVSCIBSC (sample)
Run the sample RPC sample program, DVSCIBSC. To do this, compile the COBOL member, which is located in *hlq*.SAVZSAMP data set.

Note: Use the LE370 COBOL compiler. If this compiler is not available, use the COBOL II compiler.

The PART transaction that is delivered by IBM with IMS is used by the sample program and must be installed in your IMS system.

Query syntax
The syntax of the query is:

Call DVSCIBSC

Using the ODBC CICS RPC, SDCOCIRP (sample)

Procedure

1. Compile and link the program SDCOCIRP, found in *hlq*.SAMP file.

   Note: Use the LE370 COBOL compiler. If this compiler is not available, use the COBOL II compiler.
2. When you link your CICS program, include SDCPCR, the name of the ODBC CICS RPC API Support Module as shown in the following link step of the CICS program:

```
//LKED     EXEC PGM=IEWL,REGION=&REG,
//                 PARM='&LNKPARM' ,COND=(5,LT,COB)
//SYSLIB     DD     DSN=&INDEX..SDFHLOAD,DISP=CICS load lib.
//           DD     DSN=&LE370HLQ..SCEELKED,DISP=SHRLE library (if using COBOL
for MVS)
//           DD     DISP=SHR,DSN=HLQ.LOAD=Product load lib.
//SYSLMOD    DD     DSN=&PROGLIB(&PROGRAM),DISP=SHRCICS application lib.
//SYSUT1     DD     UNIT=&WORK,DCB=BLKSIZE=1024,
//                  SPACE=(1024,(200,20))
//SYSPRINT   DD     SYSOUT=&OUTC
//SYSLIN     DD     DSN=&&LOADSET,DISP=(OLD,DELETE)
//           DD     DSN=&&COPYLINK,DISP=(OLD,DELETE)
//           DD     DDNAME=SYSIN
//*
//STEP01            PEND
//ONE        EXECDFHYITVL,PROGRAM=SDCOCIRP
//LKED.SYSIN DD *
  INCLUDE SYSLIB(SDCPCR) CICSRPC API support module
  NAME SDCOCIRP(R)
//*
```

Query syntax
The syntax of the query is:

Call CICSRPC.sdcocirp('100100')

where:

`CICSRPC` is the prefix that identifies the RPC as an ODBC CICS RPC.

`sdcocirp` is the program name in CICS that is the RPC.

'`100100`' is one or more parameters for the ODBC CICS RPC.

Monitoring and managing the CICS environment
The CICS Control Facility allows you to view and manage your CICS connections and determine the status of CICS connections and sessions.

Invoking the CICS Connections Control Facility

Procedure

1. From the Primary Option Menu, select **CICS** and press Enter.
2. Select **CICS Connections** from the **CICS Control Facility** menu and press Enter.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
F	Formats the information for the selected row.
S	Displays the CMLI control block for the selected row.
P	Prints the CMLI control block for the selected row.
R	Displays the sessions for the selected row.
I	Changes the desired status of the CICS to the INSERVICE status.
O	Changes the desired status of the CICS to the OUTSERVICE status.

Type the command to the left of the line and press Enter.

Column names
The following table provides a description and sort name (if available) for each column name on the ISPF panels.

Column name	Description	Sort name
CONNECTION NAME	The name of the connection that is defined in Data Virtualization Manager.	CONNECTION
NET NAME	The NETNAME defined in Data Virtualization Manager.	NET
GROUP NAME	The name of the group that is defined in Data Virtualization Manager.	GROUP
SECURITY NAME	The security name that is defined in the connection.	SECURITY
ACCESS METHOD	The protocol that is used for communications (EXCI).	ACCESS

Column name	**Description**	**Sort name**
IN SERVICE	The status of the service: • YES indicates the link is available for work • NO indicates the link is unavailable for work • UNKNOWN indicates the link is in transition	
TOTAL SESSIONS	The total number of sessions for the connection.	TOTAL
ACTIVE SESSIONS	The number of active sessions for the connection.	ACTIVE
PROTOCOL TYPE	The protocol being used for the connection.	PROTOCOL
QUEUE DEPTH	The number of requests waiting for a session.	QUEUE
QUEUE TIME	The amount of time the request has been waiting for a session.	TIME
LB CNCT GROUP	The name of the load balancing connection group.	LBGRP
TOT TX COUNT	The number of transactions executed.	
RUN TX COUNT	The number of transactions executed from the start of the refresh interval.	
CONVERT TO ACI	Converts CICS EXCI requests to ACI.	

Invoking the CICS Sessions Control Facility

Procedure

1. From the Primary Option Menu, select **CICS** and press Enter.
2. From the **CICS Control Facility** menu, select **CICS Sessions** and press Enter.

   This example shows the first of two panels for displaying the connections control facility. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	**Description**
F	Formats the information for the selected row.
S	Displays the CMLI control block for the selected row.
P	Prints the CMLI control block for the selected row.

Line commands	Description
I	Changes the desired status of the CICS to the INSERVICE status.
O	Changes the desired status of the CICS to the OUTSERVICE status.

Type the command to the left of the line and press Enter.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
SESSION NAME	The name of the session.	SESSION
SESSION ID	A unique identifier for each session.	ID
GROUP NAME	The name of the group that is defined in Data Virtualization Manager.	GROUP
CONNECTION NAME	The name of the connection that is defined in Data Virtualization Manager.	CONNECTION
PROTOCOL TYPE	The protocol that is used for the connection.	PROTOCOL
RECEIVE PREFIX	The prefix that is used for receive sessions.	RECEIVE
RECEIVE COUNT	The number of receive sessions to be created for the connection.	COUNT
SEND PREFIX	The prefix that is used for send sessions.	PREFIX
SEND COUNT	The number of send sessions to be created for the connection.	COUNT
IOAREALN MINIMUM	The minimum length of input/ output area.	MINIMUM
STATUS	The status of the session (IN SERVICE or OUT OF SERVICE).	STATUS
RESTART COUNT	The number of times the session has been restarted.	
TRANSACTION COUNT	The number of transactions that are executed on this session.	

IBM Data Virtualization Manager for z/OS Interface for DB2

The IBM Data Virtualization Manager for z/OS Interface for DB2 offers access to DB2-z/OS data, providing maximum performance for organizations that need to integrate DB2 data with distributed or Web applications without sacrificing flexibility, reliability, or security.

Regardless of how the data is initially represented, the IBM Data Virtualization Manager for z/OS Interface for DB2 can integrate DB2 data and stored procedures without custom coding. In addition, one Data Virtualization Manager server can access many DB2 subsystems.

The DB2 Interface Facilities option on the Primary Option Menu provides access to the Server Database Control feature. The Server Database Control application allows you to view and modify the product Server Database table. This table maps database names to entries in the Link table, which can be displayed using the Link Control application. You can associate a database name with a new host name (link) using a line command.

Database control program
The Database control program allows you to view the DB2 databases and group attachment names known to the server, and to reset the logging request queue. The entries in this table are referenced for DB2 thread collection.

Invoking the DB2 control program

Procedure

From the Primary Option Menu, select **DB2** and press Enter.

The Database Control program displays the first of two connections control facility panels. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
C	Clears the pending logging requests.
F	Formats database information for the selected row.
P	Prints the associated control block for the selected row.
S	Displays the control block for the selected row.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
SUBSYSTEM NAME	The name of the database as it will be referred to in application programs.	NAME
SUBSYSTEM TYPE	The type of database management system.	TYPE
DATABASE VERSION	The version of the database management system	VERSION
DATABASE STATUS	The status of the database management system.	STATUS
MEMBER OF GROUP	Database is a member of group attachment.	GROUP
COMPLETED REQUESTS	The number of completed requests for the database management system.	COMPLETED REQUESTS

Column name	Description	Sort name
PENDING REQUESTS	The number of pending requests for the database management system.	PENDING REQUESTS
SSCT ADDRESS	The address of the Subsystem Communication Table (SSCT) for this database management system.	SSCT ADDRESS
RIB ADDRESS	The address of the Release Information Block (RIB) for this database management system.	RIB
DB MODE	Database operational mode. Valid values are: • CM: compatibility mode. • ENFM: enable new function mode. • NFM: new function mode.	RIB

IBM Data Virtualization Manager for z/OS Interface for IMS DB: support for DBCTL

IMS support for DBCTL accesses IMS data by using DL/I data calls through the CCTL (coordinator controller). The CCTL provides communications for the DBCTL environment and consists of a subsystem that contains a database resource client (DRA).

The DBTCL (database control) is an environment that allows full-function and data entry databases (DEDBs) to be accessed from a management system.

The IMS Interface Facilities option on the Primary Option Menu provides access to the Server IMS Data Mapping Facility features.

Table 28. Server IMS Data Mapping Facility

Option	Description
Facilities	General IMS Facilities Menu
IMS Data Mapping	Create IMS Map Information
ODBA	Open Database Access Menu

Choosing a connectivity method

The IBM Data Virtualization Manager for z/OS Interface for IMS DB allows access to IMS data when used with the Data Virtualization Manager client, JDBC, or ODBC.

Using the IBM Data Virtualization Manager for z/OS Interface for IMS (CCTL/DBCTL), you can access data by using the method that is described in the following section.

SQL access to IMS DB

The IBM Data Virtualization Manager for z/OS Interface for IMS CCTL/DBCTL allows you to access IMS data by using SQL.

- Logical DBDs are not supported.
- Only the first PCB of the PSB is used.

The process of enabling access to an IMS database involves extracting database information and issuing a query. For more information, see "Using the method for SQL access to IMS DB" on page 71.

Extracting database information

You can extract information about the database from the following sources:

- IMS Database Description (DBD)
- Program Specification Block (PSB)
- Segment detail definitions

Data Virtualization Manager server maintains segment detail definitions in the Virtualization Facility. The primary segment information can be obtained from the IMS DBD for a specific database. The DBD contains segment definitions, which can be viewed as individual segment descriptions. Segment definitions contain information that describes the relationships between segments (parent/child relationships), as well as the information access path.

IMS Database Description (DBD)

To access an IMS database, the IBM Data Virtualization Manager for z/OS Interface for IMS DB/SQL requires that the Database Description (DBD) be extracted to create a DMF data mapping entry for every DBD/segment combination.

Program Specification Block (PSB)

Access to the DBD is controlled by program views, named Program Communication Blocks (PCBs). PCBs are contained in the PSBs. To enable SQL access, the PSB that contains the necessary data must be extracted to match each PCB in the PSB to DBD/segment DMF entries.

When you extract the PSB, remember the following considerations:

- SELECT, INSERT, UPDATE and DELETE operations are supported with IMS.
- Each SQL execution creates a new unit-of-work (PSB scheduling and un-scheduling) inside IMS DBCTL. Note the following points:
  - SQL SELECT operation ignores setAutoCommit() specification.
  - SQL INSERT/UPDATE/DELETE operations can honor setAutoCommit() specification; however, you must set a statement and issue a commit call with each SQL execution.
- Segment sensitivity considerations. Access is allowed to all segments contained in the first PCB for a PSB.
- Field sensitivity considerations. If field sensitivity is defined, WHERE clauses are allowed in the query.
- PCB considerations. Based on the SQL statement, the first DB PCB that has the necessary segment sensitivity and the processing options is selected for database access.

Segment detail definitions

Sometimes, database segments are not defined fully in the DBD. Segment layout detail definitions can be obtained from other sources, such as COBOL copybooks. To use segment detail definitions, they must be extracted to create DMF entries, which must be linked to the associated DBD segment.

When extracting the segment detail definitions, remember the following considerations:

- **Field Sensitivity**: If field sensitivity is defined, WHERE clauses are allowed in the query.
- **REDEFINES**: Redefinitions are used to change the information that is accessed by the IBM Data Virtualization Manager for z/OS Interface for IMS DB/SQL into a customized format, depending on how the information is to be presented.

  For example, assume PART-KEY is redefined as PART-PREFIX and PART-NUMBER:

  ```
  01 PART-REC
    03 PART-KEY PIC X(17).
    03 PART-KEY-DETAIL REDEFINES PART-KEY.
      05 PART-PREFIX            PIC X(02).
      05 PART-NUMBER            PIC X(15).
    03 FILLER
  ```

  In this case, the following SELECT statement is valid for column selection:

```
SELECT PART-PREFIX, PART-NUMBER FROM DI21PART.DFSSAM03_PARTROOT
```

- **OCCURS:** The Data Mapping Facility does not support OCCURS clauses that contain the DEPENDING ON clause. When the OCCURS clause is used, it appends a numeric suffix to the corresponding column.

  For example, if you executed the following OCCURS clause on PART-PREFIX:

  ```
  05 PART-PREFIX OCCURS 3 TIMES
  ```

  You would see the following column names:

  ```
  PART-PREFIX-1
  PART-PREFIX-2
  PART-PREFIX-3
  ```

Database information

Database information is contained in the following parts:

- DBD: DI21PART
- PSB: DFSSAM03

DI21PART and DFSSAM03 are samples to demonstrate how IMS support works. The samples represent how data shown in a hierarchical model is virtualized in tables.

Database definition (DBD)

This example is the DI21PART DBD of the PART sample database, represented in an IMS view in Figure 1 on page 66.

```
DBD NAME=DI21PART,ACCESS=(HISAM,VSAM)
DATASET DD1=DI21PART,DEVICE=3380,OVFLW=DI21PARO,
        SIZE=(2048,2048),RECORD=(678,678)
SEGM    NAME=PARTROOT,PARENT=0,BYTES=50,FREQ=250
FIELD   NAME=(PARTKEY,SEQ),TYPE=C,BYTES=17,START=1
SEGM    NAME=STANINFO,PARENT=PARTROOT,BYTES=85,FREQ=1
FIELD   NAME=(STANKEY,SEQ),TYPE=C,BYTES=2,START=1
SEGM    NAME=STOKSTAT,PARENT=PARTROOT,BYTES=160,FREQ=2
FIELD   NAME=(STOCKEY,SEQ),TYPE=C,BYTES=16,START=1
SEGM    NAME=CYCCOUNT,PARENT=STOKSTAT,BYTES=25,FREQ=1
FIELD NAME=(CYCLKEY,SEQ),TYPE=C,BYTES=2,START=1
SEGM NAME=BACKORDR,PARENT=STOKSTAT,BYTES=75,FREQ=0
FIELD NAME=(BACKKEY,SEQ),TYPE=C,BYTES=10,START=1
DBDGEN
FINISH
END
```

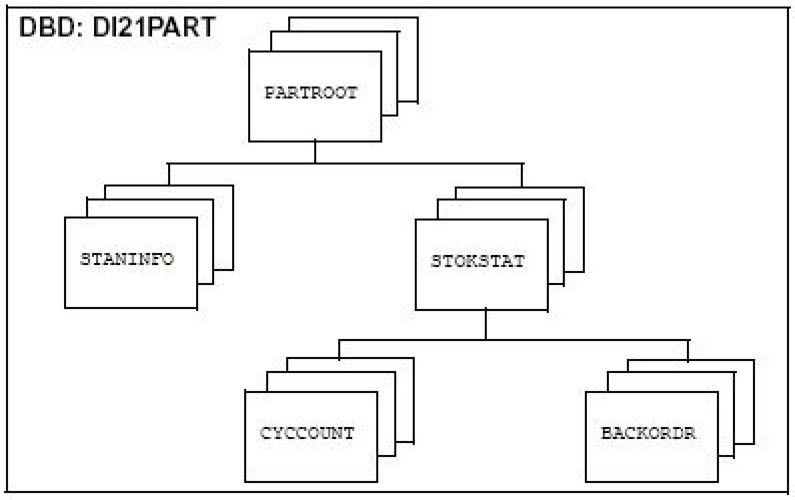


Figure 1. IMS database representation

Program Specification Block (PSB)

This example is the DFSSAM03 PSB of the PART sample database:

```
DBPCB01 PCB     TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=43
         SENSEG NAME=PARTROOT,PARENT=0
         SENSEG NAME=STANINFO,PARENT=PARTROOT
         SENSEG NAME=STOKSTAT,PARENT=PARTROOT
         SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT
         SENSEG NAME=BACKORDR,PARENT=STOKSTAT
         PSBGEN LANG=COBOL,PSBNAME=DFSSAM03
         END
```

Extracting the data
After the maps of the DBD and PSB are extracted, you can use the Data Mapping Facility to navigate through the data.

Because IMS does not maintain a catalog that describes client information for each segment, Data Virtualization Manager server maintains the information in the Data Mapping Facility. An IMS database segment map definition is created based on the SQL statement processing requirements.

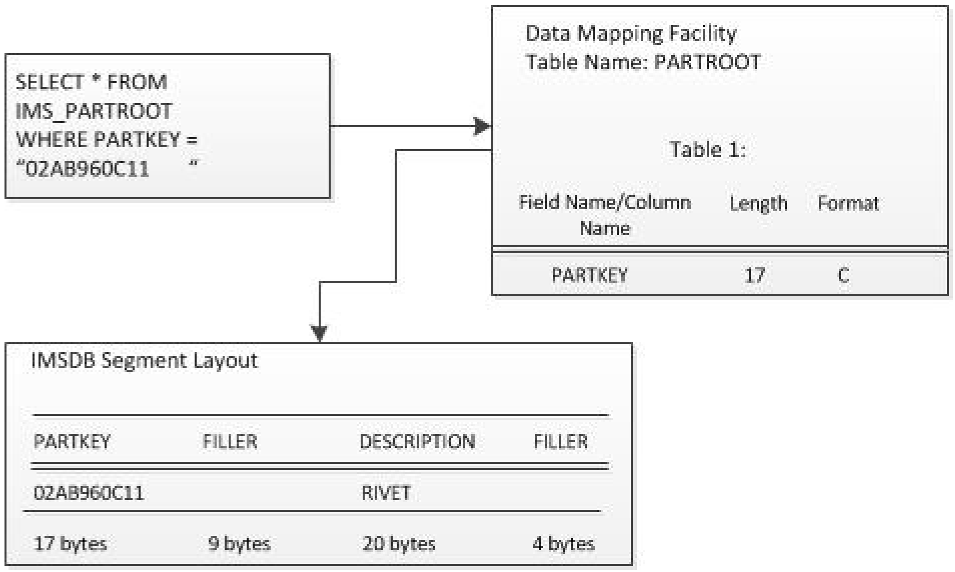


Figure 2. Using the Data Mapping Facility with the IBM Data Virtualization Manager for z/OS Interface for IMB DB/SQL

Data access paths
Data can be accessed in or across hierarchical boundaries. For the DBD shown , all of the SELECT statements that are shown in this section are valid.

The database representation of the DBD shown can be combined with the PSB and divided into specific data paths.

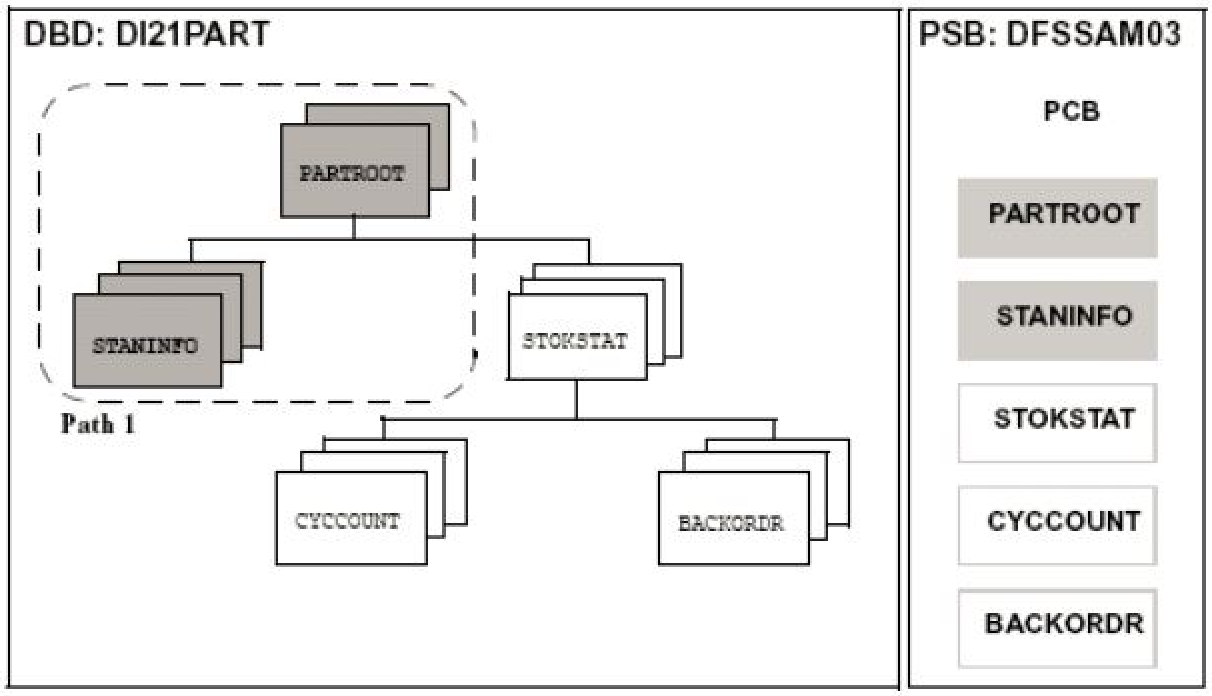


Figure 3. Data access path 1

The following SELECT statements are valid for the data access path that is shown in Figure 3:

```
SELECT * FROM IMS_PARTROOT
SELECT * FROM IMS_PARTROOT P, IMS_STANINFO C
  WHERE P.CHILD_ID = C.PARENT_ID
  AND P.PARTKEY='02AB960C11'
```

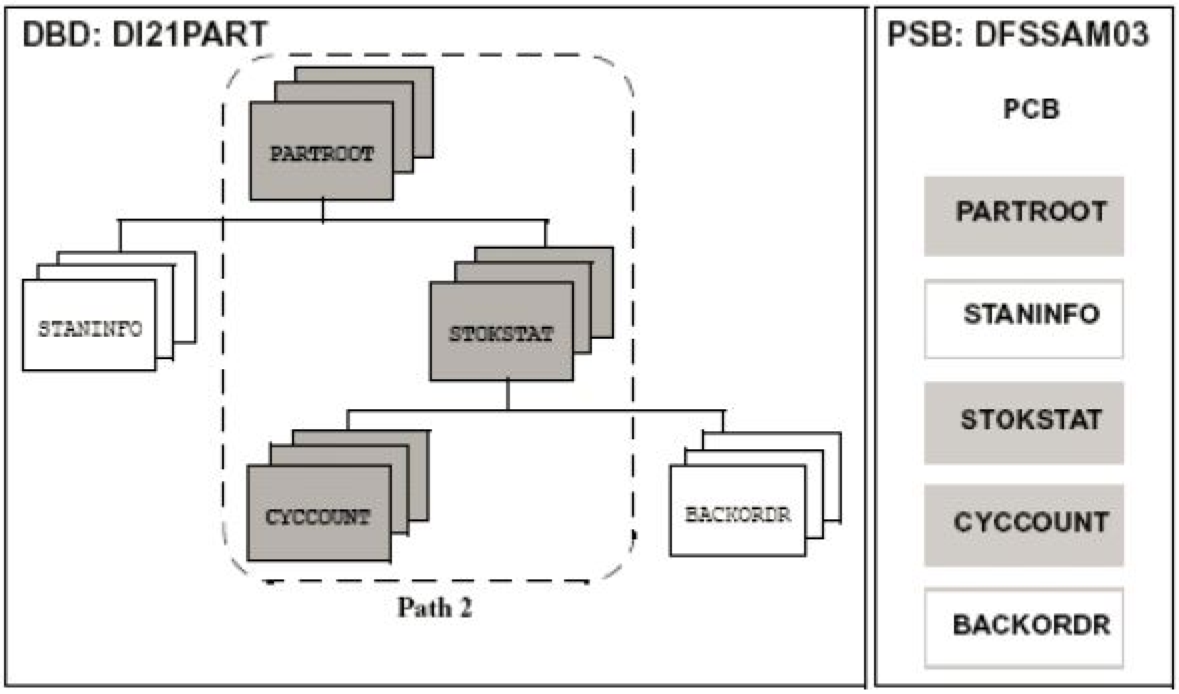


Figure 4. Data access path 2

The following SELECT statements are valid for the data access path that is shown in Figure 4:

```
SELECT * FROM IMS_PARTROOT
SELECT * FROM IMS_STOKSTAT
SELECT * FROM IMS_CYCCOUNT
SELECT * FROM IMS_PARTROOT P, IMS_STOKSTAT C
  WHERE P.CHILD_ID = C.PARENT_ID
  AND PARTKEY='02AB960C11'
SELECT * FROM IMS_PARTROOT P, IMS_STOKSTAT C1,
  IMS_CYCCOUNT C2
  WHERE P.CHILD_ID = C1.PARENT_ID
  AND C1.CHILD_ID = C2.PARENT_ID
  AND PARTKEY='02AB960C11'
```

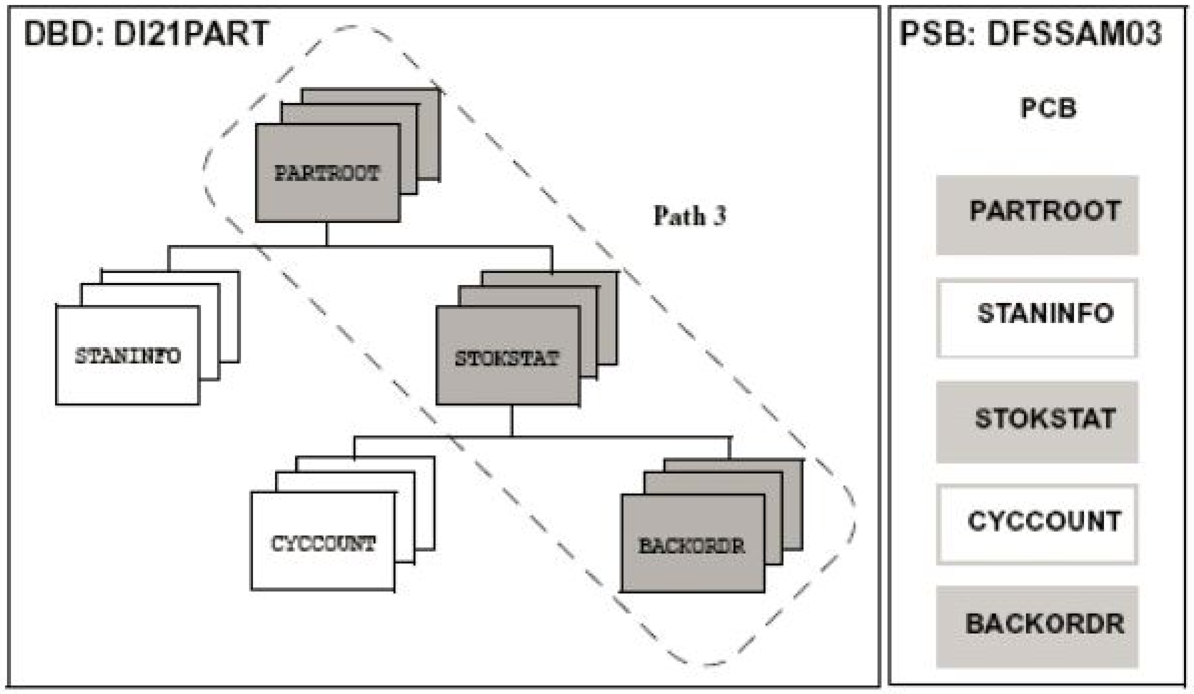


Figure 5. Data access path 3

The following SELECT statements are valid for the data access path that is shown in Figure 5:

```
SELECT * FROM IMS_PARTROOT
SELECT * FROM IMS_STOKSTAT
SELECT * FROM IMS_BACKORDR
SELECT * IMS_PARTROOT P, IMS_STOKSTAT C
  WHERE  P.CHILD_ID = C.PARENT_ID
  AND PARTKEY='02AB960C11'
SELECT * FROM IMS_PARTROOT P,
  IMS_STOKSTAT C1,IMS_BACKORDR C2
  WHERE P.CHILD_ID = C1.PARENT_ID
  AND C1.CHILD_ID = C2.PARENT_ID
  AND PARTKEY='02AB960C11'
```

The following statements are not valid because they produce a Cartesian product (or Cartesian join):

```
SELECT * FROM IMS_PARTROOT, IMS_STANINFO
SELECT * FROM IMS_PARTROOT, IMS_STOKSTAT,DI21PART.
  DFSSAM03_CYCCOUNT
```

Running a statement that produces a Cartesian product results in a `1002` error code.

Note: To select from two different tables, a WHERE clause must be specified.

Selecting data
The IBM Data Virtualization Manager for z/OS Interface for IMS DB/SQL code parses the SELECT statement, optimizes it, and processes the data by using the path that is determined by the optimizer. The optimizer examines the SELECT criteria, and combines and sorts it. It also validates the access path.

For generic selections (SELECT *), all enabled columns in the data map for the segments listed in the FROM clause are returned to the client. Selected columns can be requested from any segment in a given path.

PSB security checking
The IBM Data Virtualization Manager for z/OS Server interface for IMS DBCTL supports PSB authorization. By default, the user ID of the Data Virtualization Manager server address space, which is shared by all users, is used for authorization. If you require stricter security, you can set the parameter IMSPSBSECURITY to YES. If set to YES, the user ID of the user who attempts to schedule the PSB is used for authorization.

Creating a data map from SQL

Procedure

1. From the Data Virtualization Manager server - Primary Option Menu, select **IMS** and press Enter.
2. From the Server IMS Control Facility menu, select **IMS Data Mapping** and press Enter.
3. Select **Generate a View of an IMS/ DB DBD and Segment** from the menu and press Enter.
4. Provide the following information:
   - **Source Library Name**: The data set name and member name that contain the source code for the map you are creating.
   - **Start Field**: The field name where the map starts building.
   - **End Field**: The field name where the map stops building. If this name is not specified, the first field that is at the same level as the Start Field stops the build process.
   - **Case Sensitive**: If the Start Field or End Field are case-sensitive, set this value to Y (Yes) to preserve the case.
   - **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
   - **Use Offset Zero**: If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
   - **Convert Var to True**: Set this value to Y (Yes) to convert VAR fields to TRUE VAR fields. TRUE VAR fields have a 2-byte data length field preceding the data.
   - **Flatten Arrays**: Determines how arrays are processed. Valid values depend on the data source:
     - Flatten arrays into a single fixed table at runtime (Y)
     - Return arrays into separate tables at runtime (N)
     - Flatten arrays now (C)
   - **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task.
   - **DBD Name**: The name of the DBD for which you are creating a view.
   - **Segment Name**: The name of the Segment, from the specified DBD, for which you are creating a view.
   - **PSB Name**: The name of the PSB to use to access the specified segment.

     Note: If you leave this field BLANK, the product automatically selects a PSB name based on the SQL query.
   - **PCB Name**: The name of the PCB, from the specified PSB, to use to access the specified segment.

Note: If you leave this field BLANK, the product automatically selects a PCB name based on the SQL query.

Press Enter.

5. If either the DBD Name or Segment Name field is BLANK, the system displays a panel allowing you to choose a name from a selection list. Select a DBD Name or Segment Name from the Selection List.
6. Press Enter. If the operation is successful, the `Create Successful` message appears on the panel.

Using the method for SQL access to IMS DB

The IBM Data Virtualization Manager for z/OS Interface for IMS DB provides SQL access. Use the Data Mapping Facility to define maps. Maps are defined once, and then updated/replaced, if needed.

You can extract a map from a source by using either of the following methods:

- "Using the AVZMFPAR member" on page 71
- "Using the DMF parser"

Using the AVZMFPAR member

Use the AVZMFPAR member that is located in your *hlq*.SAVZCNTL data set as a sample JCL to virtualize DBC, PSB, and COBOL maps.

For more information about the parameters in the AVZMFPAR member, see "The batch extract member".

Using the DMF parser

Procedure

1. From the Primary Option Menu, select **IMS** and press Enter.
2. From the **IMS Control Facility** menu, select **IMS Data Mapping** and press Enter.
3. Select **Mapping Defaults** from the menu and press Enter.
4. Make sure the Parser Version option is set to `N` (New).
5. Extract by using a DBD source:
    a. Return to the **IMS Mapping Option** panel, and select the **Extract using DBD Source** option. Press Enter.
    b. Provide the following information:
        - **Source Library Name**: The data set name and member name that contain the source code for the map you are creating.
        - **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task. The map name is the DBD or PSB name. This name also is used as the member name for the map in the mapping data set, if possible.
    c. Press Enter. If the extract completes with no errors, the `Create Successful` message appears on the panel.
6. Extract the data map by using the PSB source.
    a. Return to the **IMS Mapping Options** panel and select **Extract Using PSB Source**. Press Enter.
    b. Provide the following information:
        - **Source Library Name**: The data set name and member name that contain the source code for the map you are creating.
        - **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task. The map name is the DBD or PSB name. This name also is used as the member name for the map in the mapping data set, if possible.
    c. Press Enter. If the extract completes with no errors, the `Create Successful` message appears on the panel.

7. Optional: Add segment detail definitions to the extracted DBD:
   a. Return to the **IMS Mapping Options** panel and select **Extract COBOL from listing**. Press Enter.
   b. Provide the following information:
      - **Source Library Name**: The data set name and member name that contain the source code for the map you want to create.
      - **Start Field**: The field name where the map starts building.
      - **End Field**: The field name where the map stops building. If this name is not specified, the first field that is at the same level as the Start Field stops the build process.
      - **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set, if possible.
      - **Use Offset Zero**: If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
      - **Flatten Arrays**: Determines whether arrays are flattened. Valid values depend on the product:
        - For IBM Data Virtualization Manager for z/OS SQL, you can specify `C` (COMPATIBLE) or `Y` (YES).
        - For IBM Data Virtualization Manager for z/OS Streams, you can specify `C` (COMPATIBLE) only.
        - For IBM Data Virtualization Manager for z/OS SQL 92, you can specify `C` (COMPATIBLE), `Y` (YES), or `N` (NO).

          Note: The `C` (COMPATIBLE) value is provided for backward compatibility with an older mapping architecture. When `C` is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
      - **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task.
   c. Press Enter. If the extract completes with no errors, the `Extract Successful` message appears on the panel. Both the map library and Data Virtualization Manager server contain the mapped structure definition.
8. Merge the other maps into the DBD maps to add the segment detail definitions from the COBOL listings to the DBD segments (see "Merging maps into a DBD map").
9. Display the maps to make sure they were all created successfully (see "Displaying maps").

Merging maps into a DBD map

Procedure

1. From the Primary Option Menu, select **IMS** and press Enter.
2. From the IMS Control Facility Menu, select **IMS Data Mapping** and press Enter.
3. From the panel, select **Merge Other Maps into a DBD map** and press Enter.
4. Enter information in the **DBD Map Merge Utility** panel.
   - Provide the information for the Map Data Set Library, including values for the Project, Group, Type, and Member fields (optional) for the DBD data map. Otherwise, you can use the **Other Map Data Set Name** field to specify another data set for the DBD data map.
   - To disable duplication fields, select the **Disable duplicate fields** parameter.
   - To disable FILLER fields, select the **Disable FILLER fields** parameter.

   Press Enter.
   - If you specified a member name, that member is selected and the system displays the Data Map Linkages panel.
   - If you did not specify a member name, the system displays a **Selection List** panel.
5. Select a member:
   a. From the **Selection List** panel, type one of the following commands in front of the member name:

- B: Browse the member
- E: Edit the member
- S: Select the member

Note: You can process one or multiple members.

b. Type the END command to process the members.

6. From the **Data Map Linkages** panel, in the LINK NAME column, type the names of the data maps that were extracted from the COBOL listing to link with the DBD segments. Press Enter.
7. For each DBD segment that is linked to a data map, the Data Map link established message appears in the MESSAGE column.

   Note: To force a mapping update, you must delete or leave the link name blank, and press Enter to process. After you see the `Warning: No Linked Data Map defined` message, you can rekey the link name and press Enter to pick up the revised map layout. If you performed these steps and are unable to pick up the new definition, you must perform a Map Refresh. You can also set the option Auto Refresh to Y (Yes) on the panel prior to the map extract.
8. Type the END command to process the links. The system returns to the **IMS DBD Map Links** panel. If the linking completes with no errors, the `Create Successful` message appears on the panel.
9. Return to the **IMS Mapping Options** panel and select **Map Refresh** from the menu.

   a. Press Enter for a map refresh to add your map to the map display list. If the refresh completes with no errors, the `Refresh Successful` message appears in the upper right corner of the panel.

Displaying maps
Displaying all maps is useful to make sure that maps are created correctly.

Procedure

1. Return to the **IMS Mapping Options** panel and select **Display IMS DB DBD Maps** from the menu and press Enter.

   The system displays the DBD maps. For more information about the available line commands and column descriptions, see the following sections.
2. Return to the **IMS Mapping Options** panel and select **Display IMS DB PSB Maps** from the menu and press Enter.

   For more information about the available line commands and column descriptions see the following sections.
3. Return to the **IMS Mapping Options** panel and select **Display IMS DB COBOL/PLI Extract Maps** from the menu. Press Enter.

   The system displays the PSB maps.

   These examples show the information that displays for existing data maps. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

   - **Available Commands**

     This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

     It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
D	Disables the map causing it to be unavailable for use.
E	Enables the map for use.
K	Deletes a map, also making it unavailable for use.

Line commands	Description
P	Prints the associated control block for the selected row.
S	Displays the associated control block for the selected row.
X	Displays map elements for the selected row.

- **Column names**

  The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column names	Description	Sort name
STRUCTURE NAME	The member names in the map data set.	NAME
TYPE	One of the following types of structure: – ADABAS – Input – Output – Screen – LPTBL – Header – USER	TYPE
STATUS	Status of this map (Enabled, Disabled, or Deleted)	STATUS
LANGUAGE	Language type this map was generated from (for example, Adabas, COBOL, DB2, Natural, VSAM). Determined at the time of the extract. The extracted map is independent of language type.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The date and time the map was modified. Used only for informational purposes.	DATE
USER ID	The user ID of the map creator. Used only for informational purposes.	USERID
CREATION DATASET	The data set that the map was extracted from.	DATASET

IBM Data Virtualization Manager for z/OS Interface for VSAM and Sequential files

The IBM Data Virtualization Manager for z/OS Interface for VSAM provides seamless, real-time controlled access to VSAM files, CICS-assigned KSDS VSAM files, RRDS VSAM files, and sequential files, including flat files and partitioned data sets (PDSs) as shown in the following table.

Table 29. Access to file type by interface

Interface	VSAM			QSAM
	ESDS	**KSDS/IAM**	**RRDS**	
VSAM (read-only)	YES	YES	YES	YES
VSAM CICS (read/ write)	YES	YES	YES	NO
VSAM RLS (read/ write)	YES	YES	YES	YES

Using the IBM Data Virtualization Manager for z/OS Interface for VSAM, any ODBC- or JDBC-enabled application can use standard ODBC or JDBC facilities to make SQL requests to VSAM and sequential files and return a result set. No host programming is required.

The VSAM/Sequential Interface Facilities option on the Primary Option Menu provides access to the Server VSAM/Sequential Data Mapping Facility features.

Table 30. Server VSAM/Sequential Data Mapping Facility

Option	Description
Map Defaults	Set the defaults for the mapping facility
Extract VSAM	Extract from a VSAM file
Extract Seq	Extract from a Sequential file
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps
VSAM File Control	Displays the status of VSAM files used in the system

Using the Data Mapping Facility (DMF)

Use the Data Mapping Facility (DMF) to create data maps for VSAM and sequential file access.

Creating data maps for VSAM file access

You can extract a map from a VSAM file. These instructions also apply if you are extracting a VSAM file through CICS by selecting **CICS / CICS Data Mapping / Extract VSAM** from the Primary Option Menu.

You can extract a VSAM data map by using either of the methods:

- Using the AVZMFPAR member
- Using the DMF parser

Using the AVZMFPAR member

To extract VSAM maps in batch and to extract VSAM maps with alternate indexes, run the AVZMFPAR member that is located in your *hlq*.SAVZCNTL data set as sample JCL. A compiled listing is required to perform the extract. Also, a COBOL listing with OPT(FULL) cannot be processed to produce a map. Keywords for this process define the same elements that you specify on the ISPF panels.

Note: Perform a Map Refresh before updating the display with the IBM Data Virtualization Manager for z/OS display map command.

For more information about the VSAM parameters that are located in the AVZMFPAR member, see "The batch extract member".

Using the DMF parser
To extract a VSAM data map using the DMF parser, follow these steps.

Procedure

1. From the Primary Option Menu, select **VSAM/Sequential** and press Enter.
2. From the **VSAM/Seq Data Mapping Facility** panel, select **Map Defaults** and press Enter.
3. Make sure the Parser Version is set to N (New).
4. Return to the VSAM/Seq Data Mapping Facility Display.
5. Select **Extract VSAM** from this menu and press Enter.
6. Provide the following information:
   - **Source Library Name**: The data set name and member name that contain the source code for the map being created.
   - **Start Field**: The field name where the map starts building.
   - **End Field**: The field name where the map stops building. If not specified, the first field that is at the same level as the Start Field stops the build process.
   - **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set if possible.
   - **Use Offset Zero**: If the Start Field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
   - **Flatten Arrays**: Determines whether arrays are flattened. Valid values depend on the product:
     - For IBM Data Virtualization Manager for z/OS SQL, specify C (COMPATIBLE) or Y (YES).
     - For IBM Data Virtualization Manager for z/OS Streams, specify C (COMPATIBLE) only.
     - For IBM Data Virtualization Manager for z/OS SQL 92 Engine, specify C (COMPATIBLE), Y (YES), or N (NO).

       Note: The C (COMPATIBLE) value is provided for backward compatibility with an older mapping architecture. When C is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
   - **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem.
7. Press Enter. The system displays the VSAM Extract panel.
8. Provide the following information:
   - **For read-only VSAM files allocated to the Data Virtualization Manager server address space**: In the VSAM DSN field, type the VSAM data set name (DSN) for the data that you want to access. The DSN is dynamically allocated during the execution of the query.

     Note: To create the sample VSAM file, use the sample *hlq*.SAVZCNTL(DEFSTAFF).
   - **For READ/WRITE VSAM files via CICS**:
     - The FCT for this VSAM cluster.
     - The CICS connection name, as defined in the Data Virtualization Manager server Initialization EXEC.
     - The mirror transaction name or the transaction ID, as defined in CICS.
     - The name of the Post-Read Exit and Pre-Write Exit routines if you are using the exit processing feature.
     - Type Y or N to indicate whether to use alternate indexes for this file. For this example, specify Y (Yes).

9. Press Enter. If you selected Y for alternate indexes, the VSAM Extract panel appears.
10. Specify the name of up to eight alternate indexes and press Enter. If the extract completes with no errors, the `Extract Successful` message appears on the panel.
11. Select **Map Refresh** from the **VSAM/Seq Data Mapping Facility** menu to refresh the data maps.

Using alternate indexes for a VSAM cluster
The IBM Data Virtualization Manager for z/OS Interface for VSAM supports VSAM alternate indexes by defining a data map that contains the following items:

About this task

- For read-only VSAM files allocated to the Data Virtualization Manager server address space, the data map is the path name in the base VSAM cluster.
- For read/write access to VSAM files by using CICS, the data map is the base cluster ID and an alternate index path ID as known to CICS.

The DMF allows for the same or different views in a VSAM file by changing the map name.

Procedure

1. From the Primary Option Menu, select **VSAM/Sequential** and press Enter.
2. From the VSAM/Seq Data Mapping Facility panel, select **Extract VSAM** and press Enter.
3. Provide the following information:
   - **Listing Library**: Type the information for the listing, including values for the Project, Group, Type, and Member fields. Alternatively, you can use the **Other Partitioned Data Set Containing Listing** field to specify the data set.
   - **Map Library**: Type the information for the map output data set, including values for the Project, Group, Type, and Member fields. Alternatively, you can use the **Other Partitioned Data Set To Contain Map** field to specify another data set for the map output.
4. Provide the following information in the Listing Search Criteria fields:
   - **Start Search Field**: This is used to search the listing data set for the starting point of the language-dependent data declaration. The search criteria must be unique enough to find the specific declaration to be mapped. For best results, use the fully qualified name of the declaration as it appears in the listing.
   - **End Search Field**: If this field is blank, extraction starts with the level number of the line found and continues until an equal or higher level is processed. If you enter a value in this field, extraction continues until the ending search string is found in the listing.
   - **Offset Zero**: (Y/N) Indicates whether to set the Start Search Field offset to zero, even if it is not a group level or the first definition in a group.
5. Press Enter. The system displays the VSAM Extract panel.
6. Indicate whether to use alternate indexes for this file. Specify Y (Yes) to allow the use of alternate indexes on this file.
7. Indicate whether to treat this file as an IAM file. The default is N (No); however, if the file is an IAM file, it is still treated as an IAM file.
8. Press Enter. The system displays the following panel.
9. Provide the following information:
   - For read-only VSAM files allocated to the Data Virtualization Manager server address space, the path name of the VSAM alternative index. You can add up to 10 alternative index names.
   - For read/write access to VSAM files by using CICS, the path name of the VSAM alternative index and the CICS FCT name. You can add up to 8 alternative index names.

Defining multiple VSAM logical records in the same file
If you are using the IBM Data Virtualization Manager for z/OS Interface for VSAM support of multiple logical records in the same file, you must define different views in the VSAM file. You create different maps that contain a different view for each of the logical records.

The following examples show two logical records from two different views in the same VSAM file. One view contains demographic information, and a second view contains account information. The RECORD_TYPE column specifies the view that contains the record.

Normally, a COBOL application that reads this data reads the record's content by using a record type (or view) indicator and then uses the redefinition of the record layout. If the COBOL program uses a redefine of the data area, the data map that is extracted contains the redefined columns. The application checks the content of RECORD_TYPE and uses the appropriate columns to view the data.

An alternative to this approach is to define the views in two separate data mapping definitions. Both data maps refer to the same file, but each has a different table name to distinguish its view in the VSAM data set. Using the preceding example, the data map DEMOGRAF can contain definitions for ACCOUNT_NUMBER, RECORD_TYPE, NAME, and ADDRESS. The data map ACCOUNT can contain ACCOUNT_NUMBER, RECORD_TYPE, and ACCOUNT_BALANCE. The application can issue the following queries to obtain all rows (records) in each view:

```
SELECT * FROM DEMOGRAF WHERE RECORD_TYPE = 1
SELECT * FROM ACCOUNT WHERE RECORD_TYPE = 2
```

To alternate the views, the application can run the following statements, where the &VALUE information is substituted from the previous query ACCOUNT_NUMBER column:

```
SELECT * FROM DEMOGRAPH WHERE RECORD_TYPE = 1
SELECT * FROM ACCOUNT WHERE ACCOUNT_NUMBER = "&VALUE" AND RECORD_TYPE = "2"
```

Creating data maps for sequential file access

You need to define a sequential file before you can access sequential files. You can define and extract this map by using either of the following methods:

- Using the AVZMFPAR member
- Using the DMF parser

Using the AVZMFPAR member
To extract sequential maps in batch and to extract sequential maps with alternate indexes, run the AVZMFPAR member that is located in your *hlq*.SAVZCNTL data set as sample JCL.

You still must use a compiled listing to perform the extract. Also, a COBOL listing with OPT(FULL) cannot be processed to produce a map. Keywords for this process define the same elements that you specify on the ISPF panels.

Note: You must perform a Map Refresh before it shows in the IBM Data Virtualization Manager for z/OS display map command.

For more information about the sequential parameters that are located in the AVZMFPAR member, see "The batch extract member".

Using the DMF parser

About this task

To extract a sequential data map using the DMF parser, follow these steps.

Procedure

1. From the Primary Option Menu, select **VSAM/Sequential** and press Enter.
2. From the VSAM/Seq Data Mapping Facility panel, select **Extract Seq** and press Enter.
3. Provide the following information:

- **Source Library Name**: The data set name and member name that contain the source code for the map you want to create.
- **Start Field**: The field name that is used to start building the map.
- **End Field**: The field name that is used to stop building the map. If not specified, the first field that is at the same level as the Start field stops the build process.
- **Map Name**: The name of the map in the DMF. This name also is used as the member name for the map in the mapping data set if possible.
- **Use Offset Zero**: If the Start field is not an '01' level, start the offset at zero; otherwise, the offset starts at the offset of the field in the structure.
- **Flatten Arrays**: Determines whether arrays are flattened. Valid values depend on the product:
  - For Data Virtualization Manager server SQL, specify `C` (COMPATIBLE) or `Y` (YES).
  - For Data Virtualization Manager server Streams, specify `C` (COMPATIBLE) only.
  - For Data Virtualization Manager server SQL 92 Engine, specify `C` (COMPATIBLE), `Y` (YES), or `N` (NO).

    Note: The `C` (COMPATIBLE) value is provided for backwards compatibility with an older mapping architecture. When `C` is specified, OCCURS fields are flattened in the map and OCCURS DEPENDING ON fields generate an error message.
- **Map Data Set Name**: The data set name where the map is stored. The default is the first data set in the AVZMAPP DD statement for the subsystem in the server started task.

4. Press Enter. The system displays the Sequential Extract panel.
5. Provide the following information:
   - **For flat files**: The data set name in the **Enter DSN** field.
   - **For PDSs**: The data set name in the **Enter DSN** field. In addition, if you want to create a data map that includes columns for viewing or searching the data set name and/or PDS member name, provide the following information:
     - To view the data set name, in the DSN Column Name field, type a column name that represents the data set name information.
     - To view the PDS member name, in the Member Column Name field, type a column name that represents the member name information.
     - If you want to search by the data set name or PDS member name columns, specify Y (Yes) to indicate that the columns are allowed to be used in search criteria.

       Note: If you do not specify the appropriate information to search by data set name or member name, a query returns information for all PDS members of all of data sets, without any indication of the corresponding member name or data set name.
6. Press Enter. If the extract completes with no errors, the `Extract Successful` message appears on the panel.
7. Return to the VSAM/Seq Data Mapping Facility and select **Map Refresh** to refresh the data maps.

Query syntax
The following syntax shows the query for each type of data file:

- **VSAM data (read-only)**

```
select (5) * from vsam1
```

- **VSAM for CICS data (read/write)**

```
select (5) * from filea
```

- **Sequential files**

```
select * from flatfile
```

Using a CALL statement to obtain map metadata

The IBM Data Virtualization Manager for z/OS Interface for VSAM and Sequential Files allows users to view metadata information for VSAM or sequential file data maps on the client with a simple CALL statement. The syntax of the call is:

CALL DVS_MAP('DESCRIBE','*mapname*')

where *mapname* is the name of the map.

This call returns a result set with a single column named FORMAT. The FORMAT column contains details on the fields of the map. The FORMAT column types and their SQL equivalents are shown in the following table.

FORMAT column types	SQL equivalent
CHARACTER	SQL_CHARACTER
NUMERIC	SQL_NUMERIC
DECIMAL	SQL_DECIMAL
INTEGER	SQL_INTEGER
SMALLINT	SQL_SMALLINT
FLOAT	SQL_FLOAT
DOUBLE	SQL_DOUBLE
DATE	SQL_DATE
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP
VARCHAR	SQL_VARCHAR
LONGVARCHAR	SQL_LONGVARCHAR
BINARY	SQL_BINARY
VARBINARY	SQL_VARBINARY
LONGVARBINARY	SQL_LONGVARBINARY
UNICODE	SQL_UNICODE
UNICODE_VARCHAR	SQL_UNICODE_VARCHAR
UNICODE_LONGVARCHAR	SQL_UNICODE_LONGVARCHAR

Using the Data Mapping Facility

You can use the Data Mapping Facility to set default maps to display, copy, or refresh data maps, to view individual items in a data map, to generate RPC skeletons, and to create source library definitions.

The Data Mapping option on the Primary Option Menu provides access to the Data Mapping Facility features.

Table 31. Server Data Mapping Facility

Option	Description
Map Defaults	Set the defaults for the mapping facility
Map Display	Display all map information
Map Copy	Copy maps
Map Refresh	Refresh maps

Table 31. Server Data Mapping Facility (continued)

Option	Description
Generate a RPC skeleton	Generate an RPC program from an extracted data map
VSAM File Control	Displays the status of VSAM files used in the system
Initialize Catalog	Create Data Mapping Facility maps that represent the standard product catalog tables
Source Library Management	View or create source library definitions

Setting default values for data maps

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Defaults** and press Enter.
3. Type Y (Yes) or N (No) for Auto Refresh. Press Enter.

   Y means that the storage data maps are automatically refreshed after changes.

   N requires a manual refresh by using the Map Refresh option.

   Auto Refresh can incur significant overhead if you have several changes to make and you exit after each change. Either make all changes before exiting, or turn off Auto Refresh and use the Map Refresh option when you are finished.

   If you set this value to Y, you do not need to perform a Map Refresh before the HTML generation. If you set it to N, you must perform a Map Refresh before and after the HTML generation.

Results

The Profile Saved message appears, indicating that the data set name is saved in the user profile pool.

Displaying data maps

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Display** and press Enter.

   The **Data Mapping Block** panel displays.
3. Use the available line commands to perform the appropriate functions. The following commands are available:

   - P — Prints map
   - S — Shows map
   - D — Disables map
   - E — Enables map
   - K — Deletes map
   - X — Displays map

   Type the command name and press Enter.

Results

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
STRUCTURE NAME	The member names in the map data set.	NAME
TYPE	TYPE • ADABAS • Input • Output • Screen • LPTBL • Header • USER	TYPE
STATUS	The status of the map (Enabled, Disabled, or Deleted).	STATUS
MR	Map Reduce (Yes/No)	MR
LANGUAGE	The language of the extracted map. This value is determined at the time of the extract.	LANGUAGE
AT	Attachments (OPDWs) present in the map (Yes/No)	AT
MODIFICATION DATE TIME	The date and time the map was last modified.	DATE
USERID	The user ID of the map creator. Used only for informational purposes.	USERID
NOTE	Comments	

Viewing individual data elements

About this task

To display the contents of a data map, use the following instructions.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Display** and press Enter.
   The **Data Mapping Block** panel displays.
3. Type X next to the structure to view individual data elements of that structure. Press Enter.
   The system displays the Data Elements for the structure.
4. Use the available line commands to perform the appropriate functions. Available commands:
   - P — Prints map
   - S — Shows map
   - D — Disables map
   - E — Enables map
   - C — Changes map

   Type the command name and press Enter.

Results

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column Name	Value	Description
FIELD NAME	1-50 characters	The name of the field.
COLUMN NAME	1-18 characters	The name of the column heading. During map extract, column names were created using the field names and translating any dash characters to underscores. The map editor can be used to make column names more meaningful for users.
STATUS	• Enabled • Disabled	The status of the map.
LEVEL	1-nnn	The level in relation to other elements. This is maintained for informational purposes only.
LENGTH	1-65635	The length of the data element.
FORMAT	• Char • Bin • Packed • Decimal • Date, Time • Group	The format of the data element.
OFFSET	1-65635	An offset is maintained as the relative position 0 (zero) displacement from the beginning of the structure.
PRECISION	1-65635	The element precision.
SCALE	1-65635	The element scale.
LINKED STRUCTURE	1-8 characters	The related structure name.
LINKED COLUMN	1-32 characters	The related structure column name.
FILL CHAR	1 character	The default fill character.
FILL DATA	1-200 characters	The default fill data.
ORIGINAL STATEMENT	1-80 characters	The originating statement from which the elements were extracted. For items that were entered using the editor, these are not available.

Copying data maps

Data maps may be copied, or copied then edited to create new maps.

About this task

Use the following instructions to copy data maps.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Copy** and press Enter.

   The **Move/Copy Utility** panel displays.
3. Use the available line commands to perform the appropriate functions.

   - `C` — Copy
   - `CP` — Copy and Print
   - `M` — Move
   - `MP` — Move and Print

   Type the command name and press Enter.
4. Project, Group, and Type are used for source code management. In the From ISPF Library fields, specify the following information:

   - Project
   - Group
   - Type
   - Member (if the data set is partitioned). You can perform the following actions:
     - To move, copy, or promote a single member, type the member name.
     - To move, copy, or promote all members, type * (an asterisk).
     - To request a member selection list, leave the member name blank or specify a pattern

   Alternatively, for other partitioned or sequential data sets, you can specify the From Other Partitioned or Sequential Data Set field. Type the data set name and volume serial (volume serial number).

   Note: If you do not enter a correct password for a data set that requires one, the system prompts you in standard TSO (line) mode. On TSO/TCAM systems, it may be necessary to press the CLEAR key before you respond to the password prompt. If you enter the password incorrectly or encounter other problems, you may be prompted again to enter the password until you reach a system limit of attempts.

   Press Enter.

Refreshing data maps

Refreshing a data map may be required when changes to the underlying data structure occur. When you refresh a map, the Data Mapping Facility checks the library for modifications, and then refreshes in-core map tables from the library.

About this task

Use the following instructions to refresh a data map.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Map Refresh** and press Enter.

   If the refresh is completed with no errors, the Refresh Successful message appears on the Server Mapping Facility options menu.

Generating RPC skeletons

This option generates RPC programs from an extracted data map by generating the SQLBINDCOL statements in a new PDS member by using the skeleton program that is provided in the same partitioned data set. The skeleton program contains all the language and application-specific code that is required to perform the RPC task. Substitute your information for the required keywords, and write the new specified member.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Generate an RPC skeleton** and press Enter.

   The **RPC Generation Facility** panel displays.
3. Specify the data set information for the following fields:

   - Map library
   - RPC library
   - Skeleton library

   Press Enter to generate.

Example Cobol Program

The following program is a sample of the skeleton program that gets generated. Note the commands that begin with the '@' character.

```
CBL APOST
010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. DFSSAM02.
010080 ENVIRONMENT DIVISION.
010090 CONFIGURATION SECTION.
010100 SOURCE-COMPUTER. IBM-370.
010110 OBJECT-COMPUTER. IBM-370.
010120 DATA DIVISION.
010130 WORKING-STORAGE SECTION.
COPY SBCPHD.
77 SDF-RETURN-CODE PIC S9(05) VALUE 0.
77 STATEMENT-HANDLE USAGE IS POINTER.
77 SQL-PRECISION PIC S9(5) COMP VALUE 0.
77 SQL-SCALE PIC S9(5) COMP VALUE 0.
77 SQL-COLUMN-LEN PIC S9(5) COMP VALUE 1.
77 SQL-COLUMN-NAME-LEN PIC S9(5) COMP.
77 SQL-COLUMN-NUMBER PIC S9(5) COMP.
77 SQL-COLUMN-NAME PIC X(30).
77 SQL-COLUMN-TYPE PIC S9(5) COMP.
77 ERROR-MESSAGE-AREA PIC X(256) VALUE IS SPACES.
77 TRACE-MESSAGE-AREA PIC X(256) VALUE IS SPACES.
77 STRING-PTR PIC S9(5) COMP VALUE IS 1.
77 CONNECTION-HANDLE USAGE IS POINTER.
77 ENVIRONMENT-HANDLE USAGE IS POINTER.
77 ERROR-MSG-LENGTH-AREA PIC S9(5) COMP VALUE 0.
77 NATIVE-ERROR-CODE-AREA PIC S9(5) COMP VALUE 0.
77 SQLSTATE-DATA-AREA PIC X(6) VALUE IS SPACES.
@DATABUFFER
060110 LINKAGE SECTION.
080010 PROCEDURE DIVISION.
080020 INIT.
@SQLBINDCOL BEGIN
MOVE @LENGTH TO SQL-COLUMN-LEN.
MOVE @COLUMN_NAME_LENGTH TO SQL-COLUMN-NAME-LEN.
MOVE @COLUMN_NAME TO SQL-COLUMN-NAME.
MOVE @TYPE TO SQL-COLUMN-TYPE.
MOVE @SEQ TO SQL-COLUMN-NUMBER.
MOVE @PRECISION TO SQL-PRECISION.
MOVE @SCALE TO SQL-SCALE.
CALL 'SDCPBC' USING STATEMENT-HANDLE
SQL-COLUMN-NUMBER
SQL-C-DEFAULT
SQL-COLUMN-TYPE
SQL-PRECISION
SQL-SCALE
SQL-NO-NULLS
@FIELD_NAME
SQL-COLUMN-LEN
```

```
SQL-COLUMN-NAME
SQL-COLUMN-NAME-LEN.
MOVE RETURN-CODE TO SDF-RETURN-CODE.
IF SQL-INVALID-HANDLE OR SQL-ERROR OR SQL-NO-DATA-FOUND
PERFORM 0000-ERROR-ROUTINE
END-IF.
@SQLBINDCOL END
CALL 'SDCPTH' USING STATEMENT-HANDLE SQL-THROW-DONE.
MOVE RETURN-CODE TO SDF-RETURN-CODE.
IF SQL-INVALID-HANDLE OR SQL-ERROR OR SQL-NO-DATA-FOUND
PERFORM 0000-ERROR-ROUTINE THRU 0000-ERROR-EXIT
END-IF.
080140 EXIT-RTN.
080160 GOBACK.
0000-ERROR-ROUTINE.
MOVE 256 TO SQL-PRECISION.
IF SQL-INVALID-HANDLE GO TO 0000-ERROR-EXIT.
***********************************************************
* IF AN ERROR OCCURS CALL THE SQLERROR ROUTINE
***********************************************************
CALL 'SDCPSE' USING ENVIRONMENT-HANDLE CONNECTION-HANDLE
STATEMENT-HANDLE SQLSTATE-DATA-AREA
NATIVE-ERROR-CODE-AREA
ERROR-MESSAGE-AREA
SQL-COLUMN-LEN ERROR-MSG-LENGTH-AREA.
MOVE RETURN-CODE TO WS-ODBCAPI-RETURN-CODE.
IF SQL-SUCCESS OR SQL-SUCCESS-WITH-INFO
PERFORM 0000-ERROR-DISPLAY-ROUTINE THRU
0000-ERROR-DISPLAY-EXIT.
0000-ERROR-EXIT.
0000-ERROR-DISPLAY-ROUTINE.
***********************************************************
* SEND THE ERROR MESSAGE TO THE CLIENT USING SQLRETURNSTATUS
***********************************************************
STRING 'HOST ERROR MESSAGE - ' ERROR-MESSAGE-AREA
DELIMITED BY SIZE INTO TRACE-MESSAGE-AREA WITH
POINTER STRING-PTR
END-STRING.
CALL 'SDCPRS' USING CONNECTION-HANDLE TRACE-MESSAGE-AREA
SQL-NTS NATIVE-ERROR-CODE-AREA.
0000-ERROR-DISPLAY-EXIT.
```

Program Explanation

- The following statement causes the facility to substitute the originally extracted statements in the program at the location where the statement is found:

  ```
  @DATABUFFER
  ```

- The following statements declare the beginning and ending of the SQLBINDCOL substitution. All of the statements between the begin and end are replicated for the number of ENABLED fields in the map data.

  ```
  @SQLBINDCOL BEGIN
  @SQLBINDCOL END
  ```

- The following keywords may be contained between the SQLBINDCOL BEGIN and SQLBINDCOL END statements. These keywords are substituted with the correct values for each **ENABLED** field in the data map.

  ```
  @LENGTH - the length of the field element
  @COLUMN_NAME_LENGTH - the length of the column name.
  @COLUMN_NAME - the column name used to identify the field
  @TYPE - SQL data type of column data. All DB2 SQL data types are supported
  except for graphic (DBCS) data.
  @SEQ - a sequentially assigned number for this column
  @PRECISION - the precision of the field
  @SCALE - the scale of the field
  @FIELD_NAME - the field name itself as defined in the @DATABUFFER -
  ```

Considerations

The skeleton can contain as many or as few statements as needed. It does not have to be a complete program, and you do not have to use all keywords.

For example, a skeleton member that contains the following statements generates a list of **ENABLED** field names as defined in the data map:

```
@SQLBINDCOL BEGIN
@FIELD_NAME
@SQLBINDCOL END
```

Initializing catalogs

You can create data maps that represent standard IBM Data Virtualization Manager for z/OS catalog tables. These tables are used when the call wrapper is eliminated in the ODBC and JDBC drivers. They are also used to generate the metadata information that determines access to the data.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Initialize catalog** and press Enter.

   The **Catalog Extract** panel displays.
3. Specify the map data set information:

   - Project
   - Group
   - Type

   Alternatively, you can use the **Other Map Data Set Name** field to specify the map data set.

   Press Enter to perform the catalog extract. The `Catalog Entries Defined` message is displayed on the panel.
4. Use the END command (or press F3) to return to the **Data Mapping Facility** panel.
5. Select the **Map Refresh** option. Press Enter.

   The system displays the maps. In the LANGUAGE column, those maps with the value 'CATALOG' in the language value represent IBM Data Virtualization Manager for z/OS catalog tables.

   Note: In the STRUCTURE NAME column, the following entries contain the value 'CATALOG' in the LANGUAGE column:

   - COLUMNS
   - FOREIGNK
   - PRIMARYK
   - SPECIALC
   - STATISTI
   - TABLES

Specifying catalog names on metadata calls

If you are using WebMethods, specify catalog names on metadata calls for ADABAS, IMS/SQL, VSAM, and/or CICS VSAM.

About this task

To specify a catalog name on metadata calls, perform the following:

Procedure

1. Add the parameters you want to use to the Data Virtualization Manager configuration member, AVZSIN00, using the MODIFY PARM command:

```
MODIFY PARM NAME(CATADABAS) VALUE(NULL)
MODIFY PARM NAME(CATSQLIMS) VALUE(NULL)
MODIFY PARM NAME(CATVSAM) VALUE(NULL)
MODIFY PARM NAME(CATVSAMCICS) VALUE(NULL)
MODIFY PARM NAME(POPULATECATNAME) VALUE(YES)
```

Change the value of the POPULATECATNAME parameter from NO (the default) to YES.

2. Refresh the catalog maps TABLES and COLUMNS3. You can do this using one of the following methods:
   - Use the Initialize Catalog option from the **Data Mapping Facility** panel.
   - Copy the maps from the distribution map library.

Creating source library maps

You can create new source library maps.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Server Data Mapping Facility** panel, select **Source Library Management** and press Enter.
3. From the **Data Mapping Facility** menu, select **Create Source Library Map** and press Enter.

   The **Source Library Management** panel displays.
4. Enter information for the Source Library Definitions.
   - Type the name of the list of Source Libraries.
   - Enter Y or N to specify whether to Replace an existing definition.
   - Enter information to specify the Data Set Source Library or Natural Source Library. Data Set supports all libraries except Natural.

   Press Enter.

   The system displays the **Source Library** panel and shows the new source library map.

Results

The following table describes each column name on the ISPF panels.

Column name	**Description**
NAME	The source library name.
DESCRIPTION	A description of the source library.
REPLACE	Replace an existing definition. Yes or No.
DATA SET NAME	The name of the PDS/Sequential file that contains the source code.
NATURAL LIBRARY	The name of the Natural library.
ADABAS DBID	The Adabas database ID.
ADABAS FILE	The FUSER or FDIC file number.
SERVER TYPE	The generic ACI server to run query: • B — BATCH • C — CICS

Displaying source library maps
You can view current source library maps.

Procedure

1. From the Primary Option Menu, select **Data Mapping** and press Enter.
2. From the **Data Mapping Facility** menu, select **Display Source Library Map** and press Enter.
   The **Source Library** panel displays.
3. Use the available line commands to perform the appropriate functions. Available commands:
   - P — Prints map
   - S — Shows map
   - D — Disables map
   - E — Enables map

   Type the command name and press Enter.

Chapter 4. Security

Advanced security is available for SQL, NoSQL, Events, and Services solutions. System programmers typically configure advanced security during Data Virtualization Manager server installation.

IBM Data Virtualization Manager for z/OS provides the following security features:

- Security Optimization Management (SOM) is a unique feature within the mainframe integration suite that manages and optimizes mainframe security authentication for any process that requiries authentication, such as a web service or SQL call.
- Secure Sockets Layer (SSL) for the Data Virtualization Manager server is transparently supported by the Application Transparent Transport Layer Security (AT-TLS), an IBM TCP/IP facility.
- Enterprise auditing supports the new and unique security requirements of Internet applications, while operating in the traditional enterprise computing environment. With enterprise auditing, web applications that access IBM z/OS operating system data and transactions can be used by people who do not have mainframe user IDs.
- Data Virtualization Manager server provides protection for its resources using RACF classes, CA Top Secret classes, and CA ACF2 generalized resource rules. You can run multiple instances of Data Virtualization Manager servers and either share the authorization rules or keep them separate.

Security Optimization Management (SOM)

Security Optimization Management (SOM) caches user authorization information for logon processing. SOM reduces the overhead costs that are associated with sign-on processing for all connections to Data Virtualization Manager server.

To accomplish this task, SOM saves an ACEE (accessor environment element) in a cache when the user successfully logs on to the Data Virtualization Manager server, allowing the ACEE to be reused the next time that the user logs on to the Data Virtualization Manager server. Caching ACEEs provides improved system performance by reducing logon times and by reducing overall input/output and processor consumption through the reduction of security database accesses.

Performance gains vary based on the number of logons performed. A user can invoke a client transaction (application transaction). Users running client transactions that use a unique user ID and perform logons for each transaction, see the most performance benefit. The security database stores information about the RACF, CA ACF2, or CA Top Secret configuration for users. Security database profile records are not updated and SMF records are not written when a cached ACEE is used to satisfy a logon request.

The Data Virtualization Manager server provides both basic and advanced SOM support. With advanced SOM support, the Data Virtualization Manager server can automatically expire cached ACEEs when security changes are made to the security database for a particular user ID.

Enabling basic SOM support

Basic SOM support is available for RACF, CA ACF2, and CA Top Secret.

Procedure

You can configure basic SOM support by using the `MODIFY PARM` command to set the following parameters that are located in the `AVZSIN00` configuration member:

```
if DoThis then
  do
    “MODIFY PARM NAME(SECURITYOPTIMIZATION) VALUE(YES)”
    “MODIFY PARM NAME(SECOPTRETAIN) VALUE(28800)”
    “MODIFY PARM NAME(SECOPTTARGET) VALUE(5000)”
    “MODIFY PARM NAME(SECOPTTHRESHINT) VALUE(1200)”
    “MODIFY PARM NAME(SECOPTTHRESHOLD) VALUE(10)”
    “MODIFY PARM NAME(TRACESECOPTINT) VALUE(NO)”
    "MODIFY PARM NAME(TRACESECOPTOPS) VALUE(NO)"
    "MODIFY PARM NAME(TRACESECOPTSUM) VALUE(NO)"
```

The following table lists the parameters for configuring basic SOM support:

Parameter	Description	Valid values
SECURITYOPTIMIZATION	Controls whether to cache the security environments for successful remote user logons.	**YES** (default) Caches the security environments. **NO**
SECOPTRETAIN	Specifies the amount of time, in seconds, that a cached security environment remains valid.	28800 (default value) Minimum Value: 0 Maximum Value: 86400 (24 hours)
SECOPTTARGET	Specifies the target number of security environments to keep in the user security cache. **Note:** This target number increases if there are not enough available cache entries to maintain an entry for all currently logged-on users.	5000 (default value) Minimum Value: 500 Maximum Value: 100000
SECOPTTHRESHINT	Amount of time, in seconds, that the security cache is scanned to find entries to delete. Valid values are integers that divide evenly into 3600 (one hour). For example, 60 is a valid value, but 33 is not because it does not divide evenly into 3600.	1200 (default value) Minimum Value: 60 Maximum Value: 43200 (12 hours)
SECOPTTHRESHOLD	Specifies the percentage of cache entries that are available after threshold interval process runs.	10 (default value) Minimum Value: 5 (percent) Maximum Value: 50 (percent)
TRACESECOPTINT	Controls tracing of intervals.	**YES** **NO** (default) Do not trace intervals.
TRACESECOPTOPS	Controls tracing of operations.	**YES** **NO** (default) Do not trace operations.
TRACESECOPTSUM	Controls tracing of summary and statistical information.	**YES** **NO** (default) Do not trace summary and statistical information.

Enabling advanced SOM support

Advanced SOM support detects user ID and password changes that occur in the security database while the user ID security information is cached in Data Virtualization Manager server. When this occurs, the cached security information is expired within the Data Virtualization Manager server. The existing connection, if one exists, is allowed to continue, however a new connection request requires that the user ID be re-authenticated by the security product.

About this task

Advanced SOM support consists of two RACF exit routines: the RACF common command exit, `IRREVX01`, and the RACF new password exit, `ICHPWX01`.

The `IRREVX01` exit is automatically installed each time that the server starts. This exit routine notifies SOM whenever an **`ALTUSER`**, **`CONNECT`**, **`DELUSER`**, or **`PASSWORD`** command runs.

The `ICHPWX01` exit is installed into LPALIB and requires an IPL to enable it. Existing `ICHPWX01` exit routines can be included with the Data Virtualization Manager server version of ICHPWX01. Running without this exit means that SOM does not detect password changes that are made during logon by an application other than the Data Virtualization Manager server.

The following example shows how password processing is performed before the password exit routine is installed.

- User A logs on to the Data Virtualization Manager server. User A then logs on to TSO and specifies a new password during the logon. SOM does not detect this change.
- The Data Virtualization Manager server detects the new password only when user A uses it to log on to the Data Virtualization Manager server again.

Two cases would be encountered when the exit is not installed and the user has changed the password by logging on through another application, such as TSO:

- User A logs off the Data Virtualization Manager server and logs back on using the new password. SOM detects that the password changed, expires the exiting cache entry, and then calls RACROUTE using the new password for that logon.
- User A logs off the Data Virtualization Manager server and logs back on using the old password. SOM allows the logon, if user A's cache entry has not expired, even though the password was previously changed when logging on to TSO.

You can uninstall the exits at any time using the SOM ISPF application. The RACF common command exit, `IRREVX01`, is reinstalled the next time that you start another instance of the Data Virtualization Manager server that has the Advanced SOM Support feature enabled.

To enable Advanced SOM Support for RACF:

Procedure

1. Use the `MODIFY PARM` command to set the following parameter that is located in the Data Virtualization Manager configuration member, `AVZSIN00`:

   ```
   if DoThis then
     do
       “MODIFY PARM NAME(SECOPTADVANCED) VALUE(YES)”
   ```

   The following table lists the parameters for configuring advanced security optimization:

Parameter	Description	Valid values
SECOPTADVANCED	Specifies whether SOM uses the advanced user profile change detection exits. When set to NO, SOM checks for the existence of the exits, but does not install them. When set to YES, SOM installs the dynamic security exits, if not already installed, and checks for the existence of the static security exits. This parameter must be configured before starting the server. If an instance of the Data Virtualization Manager server has the value set to YES, all servers on that LPAR are notified of the user profile changes. If a newer version is available and this parameter is set to YES. SOM replaces the exit. Advanced security features are only available for the RACF security server.	**YES** **NO** Default value is NO.

2. Install RACF New password exit, `ICHPWX01`, using the job in member `LINKPWDX` in the `hlq.SAVZCNTL` data set. Follow the instruction in the job to install the exit routine.

Using PassTickets

There is a special consideration for SOM when using PassTickets. It is inefficient to cache ACEEs for users that are authenticated using PassTickets regardless of the `Replay Protection` parameter setting.

About this task

You can tell SOM not to cache an ACEE for logons using PassTickets by setting the **`ATH.AUPWUSPT`** parameter to `1` in a `SEF ATH LOGON` rule.

Logon and logoff processing

When a client logs on to the server, SOM searches the ACEE cache for a match using the user ID and password that is supplied by the remote client. If a matching entry is found, the ACEE associated with that cache entry is used for that transaction and the counter for that cache entry is increased by one.

If an available matching entry is not found, a RACROUTE call verifies the user ID and password. After the user ID and password are verified, an entry is added to the cache for the new ACEE, and the counter for the matching cache entry is set to one.

When a client logs off, the counter for the entry is decreased by one.

ACEE retention and deletion

An ACEE (accessor environment element) is retained if a remote client task is using it, and the retention period is not expired.

The retention period is set after RACROUTE logon processing. The retention period is set to the value of the **`SECOPTRETAIN`** parameter, and can be overridden by the *ATH.AUPWAERT* variable set in a `SEF ATH LOGON RULE`.

ACEEs are deleted during interval processing or during cache steal operations. An ACEE is eligible for deletion when the responsibility count is zero, and the retention period is expired or the entry was marked expired.

An entry can be set to expired status by:

- The **KILL CACHE USER** command.
- The **EXPUSER SEF CMD** rule, which can be entered from an MVS console.
- Setting the **EXPIRESECOPTENTRIES** parameter to YES.
- Issuing a RACF **ALTUSER**, **CONNECT**, **DELUSER**, **PASSWORD**, or **REMOVE** command.
- Changing the password during logon processing.

Secure Sockets Layer (SSL)

Secure Socket Layers (SSL) is supported by the Application Transparent Transport Layer Security (AT-TLS), an IBM TCP/IP facility.

Data Virtualization Manager supports connections in the following ways:

- Ports that recognize an SSL connection and automatically enable an SSL session.
- Ports that are for secure connections that always send encrypted data.

Enabling SSL support

Before you begin

Your user ID must have READ permission for the IRR.DIGTCERT.LISTRING and IRR.DIGTCERT.LIST profiles in the RACF FACILITY class. If SSLUSERID is not specified, the Data Virtualization Manager server address space default user ID is used.

Procedure

1. Use the **MODIFY PARM** command to set the following parameters that are located in the Data Virtualization Manager configuration member, AVZSIN00:

```
“MODIFY PARM NAME(SSL) VALUE(YES)"
“MODIFY PARM NAME(SSLAUTODETECT) VALUE(NO)"
“MODIFY PARM NAME(SSLCLIENTAUTH) VALUE(LOCAL)"
“MODIFY PARM NAME(SSLCLIENTNOCERT) VALUE(ALLOW)"
“MODIFY PARM NAME(SSLUSERID) VALUE(USERID)"
```

Parameter	Description	Valid values
SSL	Enables SSL connections.	**YES** (default) SSL connections enabled. **NO**
SSLAUTODETECT (*Optional*)	Specifies whether the server automatically detects SSL connections that are sent on the port that is normally used for cleartext connections. **Note:** A separately configured SSL port accepts only SSL connections.	**YES** When set to YES, the server automatically detects SSL connections. **NO** (default) When set to NO, only cleartext connections can be handled on the cleartext port.

Parameter	Description	Valid values
SSLCLIENTAUTH	Specifies how SSL client certificates are authenticated. Valid values are NONE, LOCAL, and PASSTHRU. Configuration of SSL support for use in Data Virtualization Manager server requires that you designate the location of the certificate and keystore that the IBM-supplied SSL components use. The SSL support for the server can be configured to use a pair of native IBM SSL key database and key stash files.	**LOCAL** (default) The server requests a client certificate during the SSL connection setup handshake. Certificates that are sent by the client are authenticated by using the certificate store that is designated by other SSL startup parameters. They are either a GSK SSL key database, or a RACF keyring. **NONE** The server does not make SSL client certificate processing active and does not request client certificates. **PASSTHRU** The server requests a client certificate during the SSL connection setup handshake. Certificates that are sent by the client are not authenticated upon receipt but are available for inspection by the transaction.
SSLCLIENTNOCERT (*Optional*)	Specifies the action to take if an SSL client fails to provide a valid x501 certificate during session establishment. **Note:** The failure by the client to provide a certificate might be because of the lack of mutually trusted signing authority. Lack of a certificate does not prevent the SSL session from being established and used. **Note:** The SSL handshake at session establishment completes before application of the FAILURE action.	**ALLOW** (default) Allows the server to continue processing, ignoring failure by the client or in ability to provide a certificate. **FAIL** The server terminates its session with the client at the earliest possible opportunity.

Parameter	Description	Valid values
SSLUSERID	Specifies the user ID under which the SSL resource manager subtask operates. If not specified, the SSL resource manager operates by using the subsystem's address-space-level user ID. This user ID must be authorized to open and read the SSL private key and certificate files. Using a separate user ID for this task prevents other transaction subtasks, and the server itself, from accessing this highly confidential information.	Null

2. To set up the ports, use the **`MODIFY PARM`** command to set the following parameters that are located in the Data Virtualization Manager configuration member, `AVZSIN00`:

   Required Ports:

```
“MODIFY PARM NAME(OEPORTNUMBER) VALUE(XXXX)"
“MODIFY PARM NAME(WSOEPORT) VALUE(XXXX)"
```

   Optional Ports:

```
“MODIFY PARM NAME(OENLPORTNUMBER) VALUE(0)"
“MODIFY PARM NAME(OESSLPORTNUMBER) VALUE(0)"
“MODIFY PARM NAME(WSOEBALANCEDPORT) VALUE(0)"
“MODIFY PARM NAME(WSOESSLPORT) VALUE(0)"
```

Parameter	Description	Valid values
OEPORTNUMBER	Sets the port number that is used to LISTEN for, and ACCEPT all inbound TCP/IP sessions that should not be considered candidates for load balancing to a different Data Virtualization Manager server in the same load-balancing group. The port number should be reserved for exclusive use by the main product address space. This must be different from the main OEPORTNUMBER and the OESSLPORT number if it is used.	0 (default)
WSOEPORT	Specifies the port number that is used to listen for all inbound Services and Data Virtualization Manager studio requests.	0 (default)

Parameter	Description	Valid values
OENLPORTNUMBER (*Optional*)	Sets the port number that is used to LISTEN for, and ACCEPT all inbound TCP/IP sessions that should not be considered candidates for load balancing to a different Data Virtualization Manager server in the same load-balancing group. The port number should be reserved for exclusive use by the main product address space. This must be different from the main OEPORTNUMBER and the OESSLPORT number if it is used.	0 (default)
OESSLPORTNUMBER (*Optional*)	Sets the port number that is used to LISTEN for, and ACCEPT all inbound encrypted OE Sockets TCP/IP sessions. This port number should be reserved for use only by the main product address space. Each copy of the main product address space needs its own port number if SSL over OE Sockets is being used. There is no default value for the SSL port number if the value is not set in the initialization EXEC.	Null
WSOEBALANCEDPORT (*Optional*)	Specifies the port number that is used to listen for Services requests that can be balanced to group members.	0 (default)
WSOESSLPORT (*Optional*)	Specifies the port number that is used to listen for Services for encrypted sessions.	0 (default)

Configuring AT-TLS manually

About this task

The IBM® Configuration Assistant for z/OS® Communications Server, an optional GUI-based tool, provides a guided interface for configuring TCP/IP policy-based networking functions. You can use the Configuration Assistant to generate the Policy Agent files. You can find more information about the Configuration Assistant for z/OS in the IBM Knowledge Center.

Enterprise auditing

Enterprise auditing was created to support the new and unique security requirements of Internet applications, while operating in the traditional enterprise computing environment. With enterprise auditing, Web applications that access z/OS data and transactions can be used by people who do not have mainframe user IDs. Enterprise auditing can also be used with non-Internet applications.

The development of enterprise auditing grew from the need to replace traditional z/OS, UNIX, and NT security architecture, because the architecture could not adequately handle the larger volumes of data that is associated with Internet applications. In addition, traditional user IDs are too costly to create and administer.

Prerequisites for creating a z/OS security environment

Generic IDs can be passed to the z/OS System Authorization Facility (SAF) to create a z/OS security environment for running an RPC.

- The generic IDs must be valid host user IDs.
- The IBM Data Virtualization Manager for z/OS parameter `TLSDYNAMICUSERIDS` in the `AVZSIN00` configuration member must be set to YES.

Note: Setting `TLSDYNAMICUSERIDS` to YES affects only the SAF processing of generic IDs. All of the other features and facilities can be used even if the `TLSDYNAMICUSERIDS` is set to NO.

Using generic and extended IDs

IBM Data Virtualization Manager for z/OS implements Enterprise Auditing with a host of related new facilities. All of the facilities are based on two IDs: generic ID and extended ID.

These two IDs are provided in addition to the traditional user IDs supported by IBM Data Virtualization Manager for z/OS. They are optional and can be used either together or separately. In addition, the generic and extended ID values can be used for application debugging, logging, tracing, and auditing purposes. In many respects, they are similar to the user parameter that can be set as part of the ODBC connection initialization; however, they have the advantage that they can be set and/or reset as many times as needed for each connection.

Note: Both the generic ID and extended ID values are only transmitted over the network when they are set for the first time or when they are changed.

The generic ID and the extended ID are supported on the host side using several different mechanisms. Each of these mechanisms is optional and can be used together.

The host mechanisms are as follows:

- APIs
- SMF per-transaction recording
- Logging
- Trace browse
- Remote users

APIs

The `SQLGetInfo` function can be used in host RPCs to access (but not update) the generic ID and the extended ID. The type values for the information are as follows:

- C: `SQL_GET_GENERICID` and `SQL_GET_EXTENDEDID`
- Cobol: `SQL-GET-GENERICID` and `SQL-GET-EXTENDEDID`
- ASM: `ODSQGIGN` and `ODSQGIEX`

Both are returned as null-terminated string values.

Note:

- The output area for the generic ID should be large enough for the eight-byte string and the one-byte null terminator.
- The output area for the extended ID should be large enough for the 128-byte string and the 1-byte null terminator.

System Management Facility (SMF) Per-Transaction Recording

By setting the SMFTRANSACT parameter to YES, the SMF per-transaction recording is activated to support the generic ID and the extended ID.

Note: The extended ID area in the SM06 record has room for only the first 50 bytes of the extended ID. A new record format is provided if the entire extended ID is needed in the future.

Logging

Setting the LOGSQLSOURCE parameter to YES activates per-SQL logging. The generic ID is stored in the **GENERIC_USERID** column, and the extended ID is stored in the **EXTENDED_USERID** column.

Note: The **EXTENDED_USERID** column only has room for the first 254 bytes of the extended ID.

The logging of SQL/transactions is performed on a per-SQL basis using a DB2 table. The default table name is AVZ.SQLSOURCE; however, this default can be changed using the LOGSOURCETABLE product parameter.

Trace Browse

If a generic ID exists, it is contained in the user ID column of Trace Browse for SQL/RPC operations. If the generic ID is set to a non-blank, non-zero value, the generic ID replaces the standard user ID in Trace Browse. This information is only provided for debugging, tracking, tracing, auditing, and so on.

Note: The standard user ID is stored in Trace Browse for non-SQL/ RPC operations (such as network input/output) even if the generic ID is set. This means that both the generic ID and the standard user ID normally appear in Trace Browse for one session.

Remote users

The remote users display includes two new columns for the generic ID and the extended ID, which contain their respective values if they are set.

Host side support

The generic ID and the extended ID are supported on the host side using several different mechanisms. These mechanisms are optional and any can be used together. Several of these mechanisms are intended for application security, auditing, logging, tracing, tracking, and so on.

The host side mechanisms are installation and application specific. The host mechanisms include:

- APIs
- System Management Facility (SMF) per-transaction recording
- Logging
- Trace Browse
- Remote users

Creating a z/OS security environment

The z/OS security environment that is created by passing the generic ID to SAF is maintained during RPC execution and influences what resources the RPC can access.

Note: The generic ID z/OS security environment has no impact on SQL execution authority. The DB2 security environment is initialized when the DB2 thread is created and is not modified later.

RPC authority checking

The generic ID security environment is used to determine the following:

- If the client is allowed to run an RPC.

- If RPC authority checking has been activated by setting the Data Virtualization Manager server parameter **CHECKRPCAUTHORITY** to YES. RPC authority checking uses RACF class/entity rules or ACF2 generalized resource rules to determine if a client is authorized to run an RPC.

Note: RPC authority checking can be used with or without generic ID SAF processing and vice versa.

Caching the z/OS security environment

For performance reasons, the z/OS security environments that are created by passing generic IDs to SAF are cached. That is, each generic ID is passed to SAF only once and the z/OS security environment is cached at the address space level. This approach allows use/reuse of the generic ID security environment with negligible overhead.

Note: There is no IBM Data Virtualization Manager for z/OS Event Facility processing of LOGONs for generic IDs even if ATH rules for LOGON are enabled. The generic ID z/OS security environments are maintained in the cache until the main product address space terminates.

Security considerations

There is a possible security exposure that is associated with using generic IDs with the Data Virtualization Manager server parameter **TLSDYNAMICUSERIDS** set to YES. In this case, a z/OS security environment is created without a password. In addition, client applications are able to use the generic ID z/OS security environment without providing a password. This means that only carefully controlled applications (running inside an Application server/Web server) should be allowed to connect to a copy of IBM Data Virtualization Manager for z/OS that has the parameter **TLSDYNAMICUSERIDS** set to YES. This restriction can be enforced several ways, including LOGON ATH rules.

Note: TLSDYNAMICUSERIDS defaults to NO and can only be set to YES by using the Data Virtualization Manager configuration member. **TLSDYNAMICUSERIDS** cannot be set to YES after the main product address space initialization has completed.

Enabling enterprise auditing

Enterprise auditing is enabled by making sure the Data Virtualization Manager Server Event Facility (SEF) ATH parameter `ATH.AUPWENTL` is set to 1 in an SEF ATH LOGON rule.

About this task

The `ATH.AUPWENTL` flag is used to control whether enterprise auditing can be used. If it is not set to “1,” the client is not recognized as a secure client, and all enterprise auditing requests from that client are ignored.

This can be done by using the sample SEF ATH LOGON rule, LOGONTLS. This sample rule checks the client IP address, and if it is set to a certain value, the rule sets the ATH.AUPWENTL to “1,” thus allowing enterprise auditing to be used from this connection. The IP address to be checked may be changed to reflect your secured server.

Protected resources

System programmers typically configure advanced security during Data Virtualization Manager server customization. Data Virtualization Manager server provides protection for its resources by using RACF classes, CA Top Secret classes, and CA ACF2 generalized resource rules.

The overall RACF class (or resource type for ACF2) for Data Virtualization Manager is specified with the server parameter `RESOURCETYPE`. Classes can be shared among multiple instances of servers and either share the authorization rules or keep them separate.

Important: If the `RESOURCETYPE` parameter is not explicitly specified, the setting defaults to `NON`, which disables all product authorization checking.

When a user invokes a Data Virtualization Manager resource, the user's ID and the class of the resource are passed to the security program for authorization. The security program uses rules that you specify to determine whether to grant access to the resource.

To expedite future authorization checks of an identical request, Data Virtualization Manager server keeps the results of all security checks in protected storage.

The “look-aside” security check information is saved on a Task Control Block (TCB) basis and remains in effect until the TCB terminates. If you are initially denied access, but later have your security profile that is changed to allow access, you must exit the ISPF/SDF application to terminate its TCB. Depending on the security package, you may have to take other actions. Under ACF2, for example, you must issue the **ACFRESET** command. All security authorization events are logged in the Server Trace facility, and if access is denied, a message is produced.

The type of access you request — ADD/ALTER, READ, or UPDATE — depends on which resource you are using. The ACF2 ADD is equivalent to the RACF ALTER. See “Access requirements” on page 103 for the type of access that is required to use Data Virtualization Manager facilities.

Enabling security parameters for resource rules

To enable the security parameters, change `if DontDoThis` to `if DoThis`.

```
if DoThis then
do
  "MODIFY PARM NAME(RESOURCETYPE) VALUE(RAVZ)"
end
```

Parameter name	Parameter description	Default value
RESOURCETYPE	RESOURCE TYPE FOR RESOURCE RULES Contains the name of the security server s class (or resource type for ACF2) that is used to perform resource access authorization checks. If not explicitly specified, this parameter defaults to NON. Valid values: **NON** Disables all product authorization checking. **Important:** If you leave generalized resource checking disabled, a security exposure may exist. Anyone with a valid TSO user ID can gain access to the Data Virtualization Manager ISPF control application, where they are fully authorized to perform the functions that are provided by the interface. This assumes, however, that the user has sufficient information at hand to log on to TSO/E and then gain access to the ISPF/SDF application. ***classname*** RACF class name or ACF2 resource type. When using RACF, the corresponding class name within RACF must start with R, for example, RAVZ.	NON

List of protected resources

The following table describes the resources that are protected by the Data Virtualization Manager security mechanism.

Note: You cannot modify the resource names.

Table 32. Protected resources

Resource name	Description
ACI.aci-mapname	Access to an ACI (Advanced Communication Interface) service definition.

Table 32. Protected resources (continued)

Resource name	Description
ADA.ADABAS-file-name	Access to an Adabas file name.
ADATRACE	Authority to issue Adabas TRACE ON and TRACE OFF commands.
ADAxxxxx.FILyyyyy	Access to an Adabas file ID number.
ATHZOOM	Access to Server Trace authorization event PF4 Zoom information.
AVZ	Access to the ISPF/SDF interactive control facility.
CICSCONNECTIONS	Access to monitor and control CICS connections.
CONTROLBLOCKS	Data Virtualization Manager internal data structures.
DATABASES	Access databases that are defined to Data Virtualization Manager.
DATAMAP	Access to the Data Mapping Facility.
FILE	Access to shared files that are defined to Data Virtualization Manager.
FILETYPE	Access to the Data Virtualization Manager file-suffix/MIME-type control table.
GLOBALS	Access to global variables.
IMSLTERM	Tables correlating user IDs or TCP/IP addresses to LTERM to legacy LTERM security can be supported using an APPC interface.
LINKS	Access to communication links that are defined to Data Virtualization Manager.
PARMS	Access to the ISPF/SDF parameter display.
RPC.<rpc_name>	RPC-based security.
SEF	Access to the Event Facility dialogs.
SIS	Access to the Instrumentation Server.
TOKENS	Access to the Data Virtualization Manager tokens display.
TRACEBROWSE	Access to the Server Trace facility.
TRACEDATA	Access to all trace data, including SQL and underlying binary file trace records.
USERS	Access to the attached/remote users applications.

Access requirements

The following table provides the type of access that is required to use each Data Virtualization Manager facility.

Table 33. Data Virtualization Manager access requirements

Resources	Action	Suggested user	Access required
ADATRACE	Issuing the **ADABASTRACE ON** and **OFF** commands.	DBA, Program Products, VTAM, Operations	READ
ATHZOOM	Viewing Server Trace authorization event PF4 zoom information.	DBA, Program Products, VTAM, Operations	READ

Table 33. Data Virtualization Manager access requirements (continued)

Resources	Action	Suggested user	Access required
AVZ	Defining links using the **ADDRESS AVZ DEFINE LINK** command.	DBA, Program Products, VTAM, Operations	ADD/ALTER
CONTROLBLOCK	Using the Data Virtualization Manager command.	DBA, Program Products, VTAM, Operations	READ
CONTROLBLOCK, AVZ	Viewing product control blocks using the ISPF/SDF option AVZ.	DBA, Program Products	READ
CONTROLBLOCK, AVZ	Modifying product control blocks using a future facility.	DBA, Program Products	UPDATE
DATABASES	Viewing databases using the **ADDRESS AVZ DISPLAY DATABASE** command.	DBA, Program Products, VTAM, Operations	READ
DATABASES, AVZ	Modifying databases using the **ADDRESS AVZ MODIFY DATABASE** command.	DBA, Program Products	UPDATE
GLOBALS	Viewing global variables.	All (DBA, Program Products, Operations, Developers, End-Users)	READ
GLOBALS	Updating global variables.	DBA, Administrator, Developers	UPDATE
IMSLTERM, AVZ	Correlating user IDs or TCP/IP addresses to LTERMs.	DBA, Administrator	READ, UPDATE
LINKS	Viewing links using the **ADDRESS AVZ DISPLAY LINK** command.	DBA, Program Products, VTAM, Operations	READ
LINKS, AVZ	Modifying links using either the **ADDRESS AVZ MODIFY LINK** command.	DBA, Program Products, VTAM, Operations	UPDATE
LINKS, AVZ	Defining databases using the **ADDRESS AVZ DEFINE DATABASE** command.	DBA, Program Products	ADD/ALTER
PARMS, AVZ	Modifying the product parameters the **ADDRESS AVZ MODIFY PARM** command.	DBA, Program Products, VTAM, Operations	UPDATE
PARMS, AVZ	Viewing all Server Trace data.	DBA, Program Products, VTAM, Operations	READ

Table 33. Data Virtualization Manager access requirements (continued)

Resources	Action	Suggested user	Access required
SEF, DATAMAP	Refreshing Data Maps	DBA, Admin	READ access to SEF; UPDATE access to DATAMAP.
TRACEBROWSE, TRACEDATA, AVZ	Issuing SQL statements via AVZSPUFI.	DBA, Program Products, VTAM, Operations	READ
USERS, AVZ	Viewing remote users the **ADDRESS AVZ DISPLAY REMOTE** command.	DBA, Program Products, VTAM, Operations	READ
USERS, AVZ	Killing remote users using the ISPF/SDF option **AVZ Admin / AVZ Group**	DBA, Operations, Developers, End-Users	READ, UPDATE
USERS, AVZ	Viewing product Data Virtualization Manager parameters using the **ADDRESS AVZ DISPLAY PARM** command.	DBA, Program Products, VTAM, Operations	READ

Defining resources to RACF

Procedure

1. Use the following JCL as a model for defining a new RACF class to the RACF class descriptor table for RAVZ.

```
//STEP1 EXEC ASMHCL
//C.SYSLIB DD DSN=SYS1.MODGEN,DISP=SHR
//C.SYSIN DD *
RAVZ ICHERCDE CLASS=RAVZ,
       ID=128,
       MAXLNTH=39,
       FIRST=ALPHANUM,
       OTHER=ANY,
       POSIT=25,
       OPER=NO
     ICHERCDE
/*
//L.SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//L.SYSIN DD *
     INCLUDE SYSLMOD(ICHRRCDE)
       ORDER RAVZ
       ORDER *** Previous user-defined classes ***
       ORDER *** Previous user-defined classes ***
       ORDER ICHRRCDE
     NAME ICHRRCDE(R)
/*
```

   Restart the Data Virtualization Manager server so that RACF recognizes the new class.
2. Perform an IPL to change the RACF class descriptor table. This procedure is necessary for RACF to recognize the new class.
3. Define all RACF resource types to class RAVZ with the following command:

```
RDEFINE RAVZ CONTROLBLOCKS UACC(NONE)
```

   Repeat the `RDEFINE` command for each RACF resource type.
4. Provide access to the resource according to the following example:

```
PERMIT CONTROLBLOCKS CLASS(RAVZ) ID(USERID) ACCESS(READ)
```

Where USERID is the ID of the user to whom you want to grant READ permissions access.

If you do not want the FACILITY class to be used, the *hlq*.SAVZCNTL(AVZRADF2) member can be used as a sample for how to define the RACF class descriptor and router table.

You can edit and submit the job in *hlq*.SAVZCNTL(AVZRARES) to define and add permissions for the resource required by your site.

5. Activate the class to RACF with the following command:

```
SETROPTS CLASSACT(RAVZ)
```

What to do next
These members must be updated every time a new security resource name such as ATHZOOM or USERS is added.

Defining resources to CA Top Secret

Procedure

1. Define an entry in the RDT, as shown in the following example:

```
TSS ADDTO(RDT) RESCLASS(AVZ) RESCODE(nn)-
       ATTR(LONG,PRIV,LIB,DEFPROT,GENERIC)-
       ACLST(NONE,ALL,ALTER=1COO,UPDATE,READ)DEFACC(READ)
```

   Where *nn* is a hexadecimal code between 01 and 3F.
2. Add all the resources to an owner with the following commands:
   TSS ADDTO(owner) *AVZ*(CONTROLBLOCKS)

   Repeat this TSS ADDTO command for all resource types.
3. Permit the resources to profiles or users as follows:

```
TSS PERMIT(userid) AVZ(TRACEDATA) ACC(READ)
```

4. You can edit and submit the job in *hlq*.SAVZCNTL(AVZTSRES) to define and add permissions for the resource required by your site.

What to do next
These members must be updated every time a new security resource name such as ATHZOOM or USERS is added.

Defining resources to ACF2

Procedure

1. Define a generalized resource class named AVZ.
2. Define resource rules for each of the resource class. Member *hlq*.SAVZCNTL(AVZA2RES) can be used as an example.
3. Use the following ACF2 command to allow users access to the resource rule:

```
ACFNRULE KEY(TRACEBROWSE) TYPE(AVZ) ADD(UID(*********userid) ALLOW
```

4. You can edit and submit the job in *hlq*.SAVZCNTL(AVZA2RES) to define and add permissions for the resource required by your site:

Optional security jobs

The following table lists the jobs that are in the *hlq*.SAVZCNTL library. The jobs can be edited and submitted for the purpose that is specified in the Description column.

Table 34. Optional security jobs

Description	RACF	CA ACF2	CA Top Secret
ACI persistent connection security	AVZRAACI	AVZA2ACI	AVZTSACI
ADABAS file name or file ID	AVZRAADA	AVZA2ADA	AVZTSADA
BPEL role-based security	RACFBPEL	ACF2BPEL	
CICS transaction security	AVZRACIC	AVZA2CIC	AVZTSCIC
DB2 RRSAF security	AVZRADB2	ACF2DB2	AVZTSDB2
IDMS transaction security	AVZRAIDM	AVZA2IDM	AVZTSIDM
IMS OTMA and transaction	AVZRAIMS	AVZA2IMS	AVZTSIMS
Permissions that are required for JVM installation	AVZRAJVM	AVZA2JVM	AVZTSJVM
MQ security for Streams	AVZRAMQ	AVZA2MQ	AVZTSMQ
IBM Data Virtualization Manager for z/OS Resource security	AVZRARES	AVZA2RES	AVZTSRES
IBM Data Virtualization Manager for z/OS RPC security	AVZRARPC	AVZA2RPC	AVZTSRPC
Defining IBM Data Virtualization Manager for z/OS started task to security product	AVZRAVDB	AVZA2VDB	AVZTSVDB
RRS XA-2PC security	AVZRAXA	AVZA2XA	AVZTSXA
Streams security	AVZRASTR	AVZA2STR	AVZTSSTR

ISPF load modules

If you use TSO Command to restrict access to TSO commands, you must define the IBM Data Virtualization Manager for z/OS ISPF load modules to your security product.

Table 35. IBM Data Virtualization Manager for z/OS load modules

Load module	Description
AVZ	TSO command to invoke S__ interactive application.
AVZ2RU	Routine to invoke IBM Data Virtualization Manager for z/OS ISPF application.
AVZI	REXX Implicit Interpreter TSO Command processor.
AVZICOMP	REXX Implicit Interpreter TSO Command processor.
AVZIDB	REXX Implicit Interpreter TSO Command processor.
AVZIMEX	REXX Implicit Interpreter TSO Command processor.
AVZOB	Alias for AVZOCP.
AVZOCP	Trace Browse routine.
AVZORU	Trace Browse routine.
AVZX	REXX Implicit Interpreter TSO Command processor (Server REXX).
AVZXCOMP	REXX Implicit Interpreter TSO Command processor.
AVZXDB	REXX Implicit Interpreter TSO Command processor.
AVZXSCAN	REXX Implicit Interpreter TSO Command processor.

Table 35. IBM Data Virtualization Manager for z/OS load modules (continued)

Load module	**Description**
AVZHOCM	Host command environment for address AVZ.
AVZISCBR	Display product control blocks.
AVZSSTRU	Display product statistics.
AVZISTBR	General-purpose table display routine.
AVZSVARU	ISPF product variables display.
AVZLINK	Main product module.
AVZRXBR	Browse routine for REXX S__ line variables.
AVZRXDM	A REXX function to call new DMF parser.
AVZRXID	A REXX function for issuing commands to IDCAMS.
AVZRXIM	Initialize the REXX environment.
AVZRXLLK	Bridge REXX TO LE/370 main routine.
AVZRXPC	Product-related control block function.
AVZRXSG	REXX function for examining storage in another address space.
AVZRXST	Product-related control block function.
AVZRXTE	Terminate REXX environment.
AVZRXTK	REXX function for parsing strings into token.
AVZRXVA	REXX function for manipulating variables in a calling REXX exec.

RACF PassTickets

The RACF PassTicket can be used instead of a user logon password.

About this task

When you use a RACF PassTicket, the default application name that is passed is the three-character subsystem ID code (for example, AVZ for IBM Data Virtualization Manager for z/OS) appended with the system SMFID. This application name must match a PTKTDATA profile name for PassTicket generation and authentication to work. For example, if the system SMFID is DEV1, the application name is AVZDEV1, and you must define a PKTDATA profile for IBM Data Virtualization Manager for z/OS with the name AVZDEV1. The default application name can be changed by using the PASSTICKETAPPNAME parameter.

Also, a PTKTDATA profile name can be further qualified by RACF user ID and/or RACF connect group (for example, AVZDEV1.AVZS or AVZDEV1.SYS1.AVZS). This allows different instances of an application to have unique single sign-on keys.

For more information on defining profiles in the PTKTDATA class, see the *z/OS Security Server RACF Security Administrator's Guide*.

Defining security for RPCs

About this task

Use the following procedure if you need to restrict access to RPCs:

Procedure

1. Use the `MODIFY PARM` command to add the following parameters that are located in the `AVZSIN00` configuration member:

   ```
   "MODIFY PARM NAME(ACF/2SAFCALL) VALUE(NO)"
   "MODIFY PARM NAME(CHECKRPCAUTHORITY) VALUE(YES)"
   "MODIFY PARM NAME(RESOURCETYPE) VALUE(RAVZ)"
   "MODIFY PARM NAME(TRACEAUTHEVENTS) VALUE(YES)"
   ```

Parameter	Description	Valid values
ACF/2SAFCALL (*ACF2 Only*)	To use RPC security with ACF2, you must run a version of ACF2 that supports SAF calls.	**YES** Enables Data Virtualization Manager server to use SAF calls for Resource Rules. **NO** (default) All users are allowed to run all RPCs. The RPC can always provide its own security.
CHECKRPCAUTHORITY	Controls whether the SEF and ACF2/RACF should be used to check whether each user has the authority to run each RPC. If set to YES, the SEF and ACF2/RACF are used to verify RPC execution authority.	**YES** The SEF and ACF2/RACF is used to verify RPC execution authority. **NO** (default) All users are allowed to run all RPCs. The RPC can always provide its own security.
RESOURCETYPE	Contains the name of the security server's class (or resource type for ACF2) that is used to perform resource access authorization checks.	
TRACEAUTHEVENTS (*Optional*)	Turn on authorization event tracing (this allows you to trace the RPC security checks).	**YES** **NO** Default value is NO.

2. If you want to define all RPCs to your security product and grant RPC access to the specific users, you must edit and submit one of the following sample jobs that are located in the *hlq*.`SAVZCNTL` library.

   - AVZRARPC for RACF security
   - AVZA2RPC for CA ACF2 security
   - AVZTSRPC for CA Top Secret security

   Note: If you enable the `CHECKRPCAUTHORITY` parameter, you must define each RPC to your security product.

Information access with the TRACEDATA resource

The TRACEDATA resource controls access to information in the trace log.

About this task

The two types of information that are contained within the Data Virtualization Manager server trace log:

- SQL source statements (the real SQL source statements, as taken from database request modules or prepared strings, which may contain objects such as table names or column names).
- Binary data that underlies the trace log.

Users who have READ authority for the TRACEDATA resource and READ authority for AVZ and TRACEBROWSE can view the entire trace log. Users who do not have READ authority have only restricted access to this information.

For SQL events, if your user ID matches the user ID associated with the event, you are permitted to look at an uncensored log of the SQL event. Otherwise, you can only see a censored representation of the SQL statement. The censored version includes the SQL verb but does not include objects, such as table names or column names.

The TRACEDATA resource restricts data differently, depending on the type of event:

- SQL Events: If your user ID matches the user ID associated with the event, you are permitted to look at an uncensored log of the SQL event. Otherwise, you can only see a censored representation of the SQL statement. The censored version includes the SQL verb but does not include objects, such as table names or column names.
- Non-SQL Events: If your user ID matches the user ID associated with the event, you are permitted to see an uncensored view of the underlying binary data for event. Otherwise, you are not allowed to see the binary data at all; no data is displayed and a message is written to the terminal.

Resource security for test versions of Data Virtualization Manager server

All resource security is simulated for test versions of the Data Virtualization Manager server running in a TSO session. The z/OS security subsystem is not consulted, because a test TSO copy of the product is not authorized to perform this type of security check. All work is performed using the TSO user's existing z/OS authorizations.

In this environment, all security checks are assumed to complete successfully. If you are running test copies of the Data Virtualization Manager server under TSO, you should find this feature helpful in deploying new applications, because you can review the security checks that occur when the application is deployed in a production environment.

Configuring Adabas security

Configure security to access Adabas data at a DBID or file number level.

About this task

Securing Adabas files at a DBID or file number level requires the use of the following Data Virtualization Manager server parameters:

- RESOURCETYPE
- SQLVTRESOURCETYPE
- ADABASSECURITY

The following sample jobs for defining Adabas security-related definitions are provided in the *hlq*.SAVZCNTL library:

- AVZRAADA for RACF

  Note: When using job AVZRAADA, make the following changes for file ID security:

  ```
  RDEFINE FACILITY ADAxxxxx.FILyyyyy UACC(NONE)
  PERMIT ADAxxxxx.FILyyyyy CLASS(FACILITY) ID(<USERID>)
  ACCESS(aaaa)
  SETROPTS REFRESH RACLIST(FACILITY)
  ```

  – Change *xxxxx* to the Adabas database ID.
  – Change *yyyyy* to the Adabas file ID.

- AVZA2ADA for CA ACF2
- AVZTSADA for CA Top Secret

Procedure

1. Locate the Data Virtualization Manager configuration member. The server initialization member is shipped in data set member *hlq*.SAVZEXEC(AVZSIN00) and may have been copied to a new data set for customization in the step "Copying target libraries" in the *Installation and Customization Guide*.
2. Ensure the following settings are set in the AVZIN00 file:

```
MODIFY PARM NAME(RESOURCETYPE) VALUE(RAVZ)
MODIFY PARM NAME(SQLVTRESOURCETYPE) VALUE(RAVZ)
MODIFY PARM NAME(ADABASSECURITY) VALUE(YES)
```

Parameter name	**Parameter description**	**Value**
RESOURCETYPE	RESOURCE TYPE FOR RESOURCE RULES Specify the name of the security server's class (or resource type for ACF2) that is used to perform resource access authorization checks. When using RACF, the corresponding class name within RACF must start with R, for example, RAVZ.	For RACF: RAVZ
SQLVTRESOURCETYPE	RESOURCE TYPE FOR SQL ACCESS TO VIRTUAL TABLES Specify the name of the security server's class (or resource type for ACF2) that is used to perform authorization checks for SQL access to metadata and virtual tables in the SQL Engine. When using RACF, the corresponding class name within RACF must start with R, for example, RAVZ.	For RACF: RAVZ
ADABASSECURITY	ADABAS SECURITY ACTIVATED Set this parameter to indicate that a resource rule is to be constructed consisting of DBID and file. **Note:** Both RESOURCETYPE and SQLVTRESOURCETYPE must be set in order for ADABASSECURITY to be in effect.	YES

Configuring CICS security

The server uses the External CICS Interface (EXCI) to access CICS programs. EXCI uses the CICS multiregion operation (MRO). EXCI is subject to the same security checks as a CICS system that uses MRO to connect to another CICS system.

For more information, please refer to the *IBM CICS RACF Security Guide*.

MRO login security

Configure MRO login security so that the server started task uses a specific MRO connection.

If you start the server started task, and the **`ALLOCATE_PIPE`** request fails with `RESPONSE(SYSTEM_ERROR), REASON(IRC_LOGON_ FAILURE)`, and `subreason1= 204(decimal)`, you must set up MRO logon security for the server. The server user ID must be authorized with UPDATE authority to the `DFHAPPL.`*`net_name`* RACF profile, where *net_ name* is the name that is used in the **`NETNAME()`** parameter of the connection definition in the Data Virtualization Manager configuration file.

Since RACF does a separate check for LOGON security, you need to issue the following command:

```
RDEFINE FACILITY (DFHAPPL.netname) UACC(NONE)
PERMIT DFHAPPL.netname CLASS(FACILITY) ID(AVZS) ACCESS(UPDATE)
SETROPTS REFRESH RACLIST(FACILITY)
```

MRO bind time security

MRO bind time security ensures that the server started task is authorized to establish a connection, or bind, to a CICS region.

If you start the server started task, and the **`OPEN_PIPE`** request fails with `RESPONSE(SYSTEM_ERROR)`, `REASON(IRC_CONNECT_FAILURE)`, and `subreason1= 176(decimal)`, you must set up MRO bind time security for the server. The server user ID must have READ authority to profile `DFHAPPL.`*`applid`*, where *`applid`* is the ID for the CICS region that is specified in the **`APPLID()`** parameter of your connection definition in your Data Virtualization Manager configuration file.

If you start a server started task, and the **`ALLOCATE_ PIPE`** request fails with `RESPONSE(SYSTEM_ERROR), REASON(IRC_LOGON_FAILURE)`, and `subreason1= 204(decimal)`, you need to set up MRO logon security for the server. The server user ID must be authorized with UPDATE authority to `DFHAPPL.`*`net_name`* RACF profile, where *net_ name* is the name that is used in the **`NETNAME()`** parameter of your connection definition in your Data Virtualization Manager configuration file.

Because RACF does a separate check for LOGON security, issue the following command:

```
RDEFINE FACILITY (DFHAPPL.netname) UACC(NONE)
PERMIT DFHAPPL.netname CLASS(FACILITY) ID(AVZS) ACCESS(UPDATE)
SETROPTS REFRESH RACLIST(FACILITY)
```

Link security

The target CICS system checks link security against requests from the server using the server region's user ID.

These security checks cover the following:

- Transaction attach security (when attaching the mirror transaction).
- Resource and command security within the Server application program.

If you have transaction and program security that is enabled in your CICS region, make sure that the server user ID can access the mirror transaction and the CICS Server program. Also, if the CICS Server program accesses a protected resource, make sure the server user ID can access the same protected resource.

User security

User security applies only when ATTACHSEC(IDENTIFY) is specified on the CICS EXCI connection definition.

The user ID, which is the same user ID used for logging on to the MVS system where the server resides, are passed in the DPL request and are used to perform the same checks as link security.

Notes:

- User security can never attain more privileges than link security allows.
- If you use ACF2 and want to associate the user ID that makes the connection from the server to the CICS transaction, add the following MRO statement in the ACF2/CICS parameter file for this CICS region:

  ```
  MRO SYSID=****,FORMAT=CICS
  ```

  Where **** is the CICS connection name that is defined in CICS.

Virtual table SAF security

A single Data Virtualization Manager server environment can provide data virtualization to multiple independent tenants or application groups. The virtual table SAF (system authorization facility) security feature provides a SAF mechanism to secure virtual tables so that each tenant can only access tables authorized for members of the tenant group.

Activating this security feature will prevent using virtual table names in metadata queries (such as, **`SQLENG.TABLES`**, **`SQLENG.COLUMNS`**), as well as querying or updating application data mapped using unauthorized table names.

Server interface parameter

The `SQLVTRESOURCETYPE` parameter in the `PRODSECURITY` parameter group defines a security class name for virtual table resource checking. By default, this system parameter defaults to the value `'NON'` indicating that security checking is disabled.

When activated with a class name, the `SQLVTRESOURCETYPE` parameter will enable SAF resource checking on metadata queries (such as, **`SQLENG.TABLES`**, **`SQLENG.COLUMNS`**) as well as virtual table queries using the resource name *resource_class.table_owner.table_name* where:

- *resource_class* is the class name define for the `RESOURCETYPE` parameter in the `PRODSECURITY` parameter group (for example, RAVZ)
- *table_owner* is the `SQL TABLE OWNER NAME (SQLENGTABLEOWNER)` as defined in the `PRODSQL` parameter group (for example: 'DVSQL')
- *table_name* is the map (or virtual table) name as defined in the map data set

For improved performance in SAF calls, `RACROUTE REQUEST=FASTAUTH` provides general resource checking. A separate `INTRNLONLY` parameter named `'DISABLE FASTAUTH SECURITY CHECKS'` disables use of FASTAUTH if security problems are encountered. Disabling FASTAUTH will switch to `RACROUTE REQUEST=AUTH` checking on all resource rules which can degrade query performance on metadata tables.

When securing metadata tables, READ access is required to query rows in the following tables.

- `SQLENG.COLUMNS`
- `SQLENG.COLUMNPRIVS`
- `SQLENG.ERRORMSGS`
- `SQLENG.FOREIGNKEYS`
- `SQLENG.PRIMARYKEYS`
- `SQLENG.ROUTINES`
- `SQLENG.SPECIALCOLS`
- `SQLENG.STATISTICS`
- `SQLENG.TABLES`
- `SQLENG.TABLEPRIVS`

Securing tables using the generic profile `SQLENG.*` is also an option if preferred.

Securing specific virtual tables is also required when activating this feature. Securing virtual tables by specific or generic rules activates two security checks:

1. When querying metadata tables (`SQLENG.*`), users must minimally have READ access to the virtual tables in order for rows related to a table to be returned. In this case, there are no errors returned. Instead, the information about a specific table is omitted from the result set and the user has no indication that the table exists.
2. When querying virtual tables, the user must have READ access to each table in the `SQL SELECT` statement and UPDATE access to any table that is the target of an `SQL INSERT`, `UPDATE`, or `DELETE` statement.

Restrictions and Considerations

Virtual table authorization checking is built on general resource checking and is impacted by the following product parameter in the PRODSECURITY group:

- ALLOWUNPROT – The ALLOWUNPROT parameter allows access to unprotected resources. When set to YES, this parameter allows access to resource names that have no matching resource definition in the SAF database. ALLOWUNPROT should be set to NO to insure resource rules are correctly processed.

  Note: ALLOWUNPROT=NO will automatically activate numerous resource checks unrelated to this feature.

The *table_owner.map_name* resource name is internally restricted to 44 bytes. While internal map names larger than 44 bytes are still allowed, resource checking will only pass the first 44 bytes of the *table_owner.map_name* string in the SAF call for validation. Generic resource rules will be necessary if map names exceed this limitation.

Because all maps are limited to a single table owner as defined in the SQLENGTABLEOWNER system parameter, users should consider a standard prefix for all map names they want to secure for application groups. This simple generic resource rules can be defined to protect these names. For example, if the SQLENGTABLEOWNER is configured as 'DVSQL' and an application group uses AG01 as a prefix on all table names, a generic resource 'DVSQL.AG01*' will control access to all tables starting with AG01 as a map name.

All SQL queries are automatically secured when this feature is activated. This means that resource rules must exist to allow READ access to the metadata tables SQLENG.*.

This feature is limited to SQL access to virtual tables. Users authorized to create tables can create tables which may not be accessible due to SQL access rules implemented using this feature.

Chapter 5. Performance

The Data Virtualization Manager server provides a number of features to enhance performance.

- Workload management – Using the IBM Workload Manager for z/OS, you can define performance goals and priorities. The system matches its resources to the work and determines whether goals are being met by monitoring and adapting its processing.
- Multiple servers – Running separate Data Virtualization Manager servers addresses server needs for testing or for distributing your workload.
- Load balancing – With load balancing, inbound connections are automatically directed to server instance that has the most available resources.
- CICS failover – This feature allows you to set up an alternate CICS ID for each CICS connection, so that if access to a primary CICS connection fails, a hot failover is performed to the alternate CICS region.
- System resource management – Several system resources, including block fetch, time limit alerts, and session failure detection maintain response times within pre-established services levels.
- Virtual Connection Facility – This feature increases the number of client connections possible.
- Enterprise transaction support – The IBM Data Virtualization Manager for z/OS Enterprise supports various two-phase commit protocols, including the Resource Recovery Services attachment facility (RRSAF).

Workload Manager (WLM)

Using the IBM Workload Manager for z/OS, you can define performance goals and assign a level of importance to each goal in business terms. The system matches its resources to the work and determines whether goals are being met by monitoring and adapting its processing. This allows you to make the best possible use of the server's resources, while achieving the best possible response times.

Goals are specified for the WLM services in IBM Data Virtualization Manager for z/OS in the same way they are specified for z/OS-managed work, by associating work with a service class. The assigned service class informs the operating system about the performance goal and importance level that is associated with the work, as well as the address spaces involved in processing the work request.

Support for the Workload Manager (WLM) is available for the SQL data access. For information about planning for and using workload management, refer to the IBM Knowledge Center for the *MVS Planning: Workload Management* and *MVS Workload Management Services* documents.

WLM enclaves

To facilitate implementation of transaction management, WLM uses enclaves. An enclave is a group of one or more logically related z/OS task control blocks (TCB) and service request blocks (SRB) that manage the work in entities.

Using enclaves provides the following benefits:

- Work running in enclave SRBs can be offloaded to a zIIP processor. The Data Virtualization Manager server runs in enclave SRB mode, when possible, to allow CPU offloading.
- The resources that are used to process the transaction can be accounted to the transaction rather than to the address space in which the transaction runs. Service class performance goals are inherited by the enclave.

The Data Virtualization Manager server establishes a logical dispatchable unit (LDU) for each process and thread in its address space. This LDU consists of a TCB/SRB pair that is dispatched in SRB mode in a WLM enclave, if possible, switching to TCB mode only if required by system or database interfaces. The SRB mode execution is eligible for offloading to a zIIP based on the definitions in the WLM service policy.

During installation, the Data Virtualization Manager server establishes two long-running enclaves. One is the service class `AVZ_SCNM`, and the other is the service class `AVZ_SCHI`. Dispatchable units join these enclaves as appropriate. Unique enclaves are created as needed for the processes and threads for SQL data access.

Configuring Workload Manager (WLM)

You use WLM to define performance goals and assign a level of importance to each goal in business terms.

The system then matches its resources to the work, as well as monitors the goals and makes necessary processing adoptions accordingly.

This section explains several ways that you can configure WLM support and provides the definitions that are required to use the support.

WLM definitions

A service definition is the name that is given to the combination of service policies, workloads, service classes, resource groups, classification rules, and application environments. It is based on the performance objectives in a service level agreement (SLA). The following is a list of WLM definitions:

Workload

A named group of work, or service classes, that is reported as a unit.

Service Class

A named group of work that has similar performance goals, resource requirements, or importance. In the service class, you assign each goal and its relative importance, and associate the service class with a specific workload and resource group. IBM Data Virtualization Manager for z/OS requires the following service classes.

- `AVZ_SCHI ZIIPCLASS=AVZ` High priority. This service class is for IBM Data Virtualization Manager for z/OS critical work. Assign this class goal as close to SYSSTC as possible.
- `AVZ_SCNM ZIIPCLASS=AVZ` Normal work. This class is for IBM Data Virtualization Manager for z/OS administrative work. Assign this class the same goals as those used for DB2 master or the IMS control region.
- `AVZ_SCTX ZIIPCLASS=AVZ` Client work. This service class is for client requests. Assign this class the same goals as those supporting the data source. This would most likely be the CICS, IMS/TM, or DB2 WLM address space.

Classification Rules

A classification rule maps work coming into the system to a specific service class and report class. A classification is based on the subsystem type and work qualifiers in the subsystem type. The work qualifiers define and associate service classes to the type of work.

Report Class

A named group of work that is for reporting purposes only. Use report classes to distinguish among types of work that run in the same service class.

Providing WLM definitions via Data Virtualization Manager

Before you begin

Before you start this procedure, it is important to understand the following requirements:

- Data Virtualization Manager must have proper access to the MVSADMIN.WLM.POLICY resource.
- Your user ID must have UPDATE access or the following error occurs:

```
*SDx0038S INSTALL OF WLM SERVICE DEFINITION FAILED, RC=X'0000000C',
REASON=X'xVy0039s', DETECTED AT OPINWM+X'FFC3BF06'
```

- Your user ID for starting the server must have READ access or the following error occurs:

```
SDx3269I WLM administration userid xxxxxxxx logged on to system
```

```
xVy0037E WLM EXTRACT SERVICE DEFINITION FAILED, RC=X'00000004', DETECTED AT
OPINWM+X'00000B02'
```

Procedure

1. Add the following statements to your `AVZSIN00` configuration member:

```
If DoThis then
  do
    “MODIFY PARM NAME(WLMFORCEPOLICY) VALUE(YES)”
    “MODIFY PARM NAME(WLMTRANNAME) VALUE(APPLNAME)”
    “MODIFY PARM NAME(WLMUSERID) VALUE(AVZS)”
End
```

The following table lists the parameters for WLM definitions:

Parameter	Description	Valid values
WLMFORCEPOLICY	Controls whether the Data Virtualization Manager server enforces service policy requirements.	**YES** The server initialization examines the active policy for required elements and terminates if the elements do not exist and the server is not allowed to add them. The server also examines the policy anytime it is refreshed, and shuts down the server if the new policy is not in compliance with server requirements. **NO** (default) The Data Virtualization Manager server checks the policy for required definitions, and issues an error message if the subsystem type (default AVZ, set by WLMSUBSYSTEM) is not defined in the policy. The server is allowed to initialize, and is not shut down for any policy changes.

Parameter	Description	Valid values
WLMTRANNAME (*optional*)	Specifies which value is used as the transaction name when classifying the Data Virtualization Manager server transactions.	**APPLNAME** (default) The application name set in the client ODBC data source is used as the transaction name. **MODNAME** The name of the application that uses the client ODBC driver is used as the transaction name. **INTNAME** The client application executable internal name is used as the transaction name.
WLMUSERID (*optional*)	Specifies a highly privileged user ID under which WLM administration functions are performed. This user ID must be authorized to update the MVSADMIN.WLM.POLICY resource. If WLMUSERID is not specified, the server subsystem ID is used for WLM policy administration.	AVZS (default subsystem ID)

2. Enter WLM from the ISPF/PDF option 6 panel to log on to the IBM TSO/ISPF WLM administration tool.
3. Extract and save a copy of the current service definition. This is for backup purposes only.
4. Optional: Update the `AVZSIN00` configuration member with a valid `WLMUSERID`.
5. Start Data Virtualization Manager server.

   Upon startup, Data Virtualization Manager:

   - Examines the current WLM service policy for the required elements. If the active policy contains the required elements, initialization continues. If the required elements are not found, Data Virtualization Manager messages xDy0706I, and xDy0707I are issued for each missing element.

```
xDy0706I DATA VIRTUALIZATION SERVER AVZS requires the following elements
 missing from WLM Service Policy active_policy_name

xVy0707I Type: WORKLOAD, Data Virtualization Parameter: WLMWORKLOAD,
Value: AVZ_WKLD

xVy0707I Type: SUBSYSTEM, Data Virtualization Parameter: WLMSUBSYSTEM,
Value: AVZ

xVy0707I Type: SERVICE CLASS, Data Virtualization Parameter:
WLMSERVICECLASS, Value: AVZ_SCNM

xVy0707I Type: SERVICE CLASS, Data Virtualization Parameter:
WLMHISERVICECLASS, Value: AVZ_SCHI

xVy0707I Type: SERVICE CLASS, Data Virtualization Parameter:
WLMTXSERVICECLASS, Value: AVZ_SCTX
xVy0707I Type: REPORT CLASS, Data Virtualization Parameter:
 WLMP1REPORTCLASS, Value: AVZ_RCP1

xVy0707I Type: REPORT CLASS, Data Virtualization Parameter:
WLMP2REPORTCLASS Value: AVZ_RCP2

xVy0707I Type: REPORT CLASS, Data Virtualization Parameter:
WLMP3REPORTCLASS, Value: AVZ_RCP3
```

```
xVy0707I Type: CLASSIFICATION RULE, Data Virtualization Parameter:
WLMTRANSACTION, Value: AVZ_TNNM

xVy0707I Type: CLASSIFICATION RULE, Data Virtualization Parameter:
WLMHITRANSACTION, Value: AVZ_TNHI

xVy0707I Type: CLASSIFICATION RULE, Data Virtualization Parameter:
WLMTXTRANSACTION, Value: AVZ_TNTX
```

- Data Virtualization Manager then examines the WLM service definition for the required elements. If `WLMFORCEPOLICY` is set to NO, the following actions are skipped. If `WLMFORCEPOLICY` is set to YES, the following actions are enforced. The default is NO.

  Action 1: If the required elements are found in the service definition, a WTOR is issued, requesting permission to activate the current service policy. If the current policy is no longer in the service definition, the user is asked to select one of the policies in the service definition for activation.

  ```
  *nn xVy0719R Reply 'GO' to activate Policy
  service_policy_name, or 'CANCEL' to terminate
  Server initialization
  ```

  If you reply with CANCEL, the Data Virtualization Manager server shuts down.

  If you reply with GO, the Data Virtualization Manager server automatically activates your WLM Policy *service_policy_name*, and you should see the following message in the system log:

  ```
  IWM001I WORKLOAD MANAGEMENT POLICY service_policy_name NOW IN EFFECT
  ```

- **Action 2**: If the required elements are not found in the service definition, the Server issues message xDy0706I, and then message xDy0707I for each missing element.

  ```
  xDy0706I DATA VIRTUALIZATION SERVER AVZS requires the following elements
  missing from WLM Service Definition service_definition_name.

  xVy0707I Type: WORKLOAD, Data Virtualization Parameter: WLMWORKLOAD,
  Value: AVZ_WKLD

  xVy0707I Type: SUBSYSTEM, Data Virtualization Parameter: WLMSUBSYSTEM,
  Value: AVZ

  xVy0707I Type: SERVICE CLASS, Data Virtualization Parameter:
  WLMSERVICECLASS, Value: AVZ_SCNM

  xVy0707I Type: SERVICE CLASS, Data Virtualization Parameter:
  WLMHISERVICECLASS, Value: AVZ_SCHI

  xVy0707I Type: SERVICE CLASS, Data Virtualization Parameter:
  WLMTXSERVICECLASS, Value: AVZ_SCTX

  xVy0707I Type: REPORT CLASS, Data Virtualization Parameter:
  WLMP1REPORTCLASS, Value: AVZ_RCP1

  xVy0707I Type: REPORT CLASS, Data Virtualization Parameter:
  WLMP2REPORTCLASS, Value: AVZ_RCP2

  xVy0707I Type: REPORT CLASS, Data Virtualization Parameter:
  WLMP3REPORTCLASS, Value: AVZ_RCP3

  xVy0707I Type: CLASSIFICATION RULE, Data Virtualization Parameter:
  WLMTRANSACTION, Value: AVZ_TNNM

  xVy0707I Type: CLASSIFICATION RULE, Data Virtualization Parameter:
  WLMHITRANSACTION, Value: AVZ_TNHI

  xVy0707I Type: CLASSIFICATION RULE, Data Virtualization Parameter:
  WLMTXTRANSACTION, Value: AVZ_TNTX
  ```

  The preceding messages are followed by a WTOR requesting permission to update the service definition.

  ```
  *nn xVy0708R Reply 'GO' to update the WLM Service Definition, or
  'CANCEL' to terminate server initialization
  ```

If you reply with CANCEL, Data Virtualization Manager server shuts down.

If you reply with GO, the Data Virtualization Manager server automatically makes the proper WLM updates to your WLM policy definition. At the conclusion of the update process, you receive the following message.

```
xDy0709I WLM Service Definition service_definition_name has been updated
with required elements
```

Action 3: A separate WTOR message is presented to activate the policy.

```
*nn xVy0719R Reply 'GO' to activate Policy service_policy_name, or
'CANCEL' to terminate server initialization
```

If you reply with CANCEL, Data Virtualization Manager server shuts down. The user can use the TSO/ISPF WLM administration dialog to extract the service definition and review the additions that are made by the Data Virtualization Manager server.

If you reply with GO, the Data Virtualization Manager server automatically activates your WLM policy *service_policy_name*, and you see the following message:

```
IWM001I WORKLOAD MANAGEMENT POLICY service_policy_name NOW IN EFFECT
```

Note: After the WLM service policy is activated, if you change any IBM Data Virtualization Manager for z/OS required WLM element in the service definition to an invalid value and activate a service policy, all servers requiring the now invalid definition shut down.

Note: You should have a backup of your existing WLM service policy definitions.

Providing WLM definitions manually

About this task

If you want to manually define the required WLM definitions rather than have the Data Virtualization Manager server automatically install them at startup time, take the following steps:

Procedure

1. Start the WLM administration tool. The IBM TSO/ISPF WLM administration tool is used in the following examples. Other administrative tools can also be used.
   a) Enter **WLM** from the ISPF/PDF option 6 panel to log on to the IBM TSO/ISPF WLM administration tool.
   b) Select **Option 2 Extract Definition from WLM Couple Data Set** from the Choose Service Definition box.
2. Define the workloads.
   a) Select **Option 2 Workloads**. Press Enter.
      WLM displays the Workload Selection List panel.
   b) Create workload AVZ_WKLD.
3. Define the service classes.
   a) Select **Option 4 Service Classes**. Press Enter.
      WLM displays the Service Class Selection List panel.
   b) Here you create the following service classes:
      - `AVZ_SCHI ZIIPCLASS=AVZ` IBM Data Virtualization Manager for z/OS high priority
      - `AVZ_SCNM ZIIPCLASS=AVZ` IBM Data Virtualization Manager for z/OS normal work
      - `AVZ_SCTX ZIIPCLASS=AVZ` IBM Data Virtualization Manager for z/OS client work

      Note:

      - Do not change service class names.

- `ZIIPCLASS=AVZ` is a required keyword in the description.
- The values that are shown for service class goals are default values that you can modify.

4. Define subsystem type AVZ and its classification rules.
   a) Select **Option 6 Classification Rules**. Press Enter.
   WLM displays the Subsystem Type Selection List for Rules panel.
   b) Define subsystem type AVZ and associated classification rules.
5. Define the report classes.
   a) Select **Option 7 Report Classes**. Press Enter.
   WLM displays the Report Class Selection List panel.
   b) In this panel, create the following report classes:
   - AVZ_RCP1 D*nnnn* P*nnn* PERIOD 1
   - AVZ_RCP2 D*nnnn* P*nnn* PERIOD 2
   - AVZ_RCP3 P100 PERIOD 3

   Note:

   - Do not change report class names.
   - The terms in the report class descriptions are used to provide CPU offload criteria for Data Virtualization Manager server work as follows:

     D*nnnn*: The number of service units during which the dispatchable units are in the associated period while eligible for offloading to the zIIP processor.

     P*nnn*: The percentage of time in the associated period that Data Virtualization Manager server tries to offload work to the zIIP processor.
6. Activate a Service Policy.
   a) Select **Option 3 Activate Service Policy** from the **Utilities** drop-down menu on the panel.
   b) Follow directions to activate a policy.

Using the WLM Administration Tool

Procedure

1. Enter the following command to start the IBM TSO ISPF administration tool:

```
TSO WLM
```

2. Follow all prompts until the **Choose Service Definition** panel is displayed.
3. Type 2 to select the **Extract definition from WLM couple data set** option.
4. Press **ENTER**. The **WLM Definition** panel appears. You can select the option for the task that you want to perform.

Workload Manager definitions

During initialization, Data Virtualization Manager server connects the server address space to the WLM and ensures that WLM elements are in the current active service policy.

Table 36. WLM Element Types

WLM Element Type	Server Parameter	Default Value
Workload	WLMWORKLOAD	AVZ_WKLD
Subsystem	WLMSUBSYSTEM	AVZ
Service Class	WLMSERVICECLASS	AVZ_SCNM

Table 36. WLM Element Types (continued)

WLM Element Type	Server Parameter	Default Value
Service Class	WLMHISERVICECLASS	AVZ_SCHI
Service Class	WLMTXSERVICECLASS	AVZ_SCTX
Classification Rule	WLMTRANSACTION	AVZ_TNNM
Classification Rule	WLMHITRANSACTION	AVZ_TNHI
Classification Rule	WLMTXTRANSACTION	AVZ_TNTX
Report Class	WLMP1REPORTCLASS	AVZ_RCP1
Report Class	WLMP2REPORTCLASS	AVZ_RCP2
Report Class	WLMP3REPORTCLASS	AVZ_RCP3

Modifying the workload

The workload, `AVZ_WKLD`, is required by the Data Virtualization Manager server.

About this task

To modify the IBM Data Virtualization Manager for z/OS workload definition:

Procedure

Select the **Workloads** option from the **WLM Definition** panel (see "Using the WLM Administration Tool"). Press Enter.

The system displays the **Modify a Workload** panel.

Results

Note: You can change the **Workload Name** field by using the `WLMWORKLOAD` parameter, which is located in the Data Virtualization Manager configuration member, `AVZSIN00`. Do not change this name unless instructed to do so by IBM Software Support.

Modifying a service class definition

Before you begin

For details about setting up service class definitions, refer to the IBM Knowledge Center for the *MVS Planning: Workload Management* and *MVS Workload Management Services* documents.

- The `AVZ_SCHI` service class is used for high importance server work, such as management tasks of short duration that should not be interrupted, establishing a new thread for a new transaction.
- `AVZ_SCNM` is the default service class for all work that is not explicitly classified, except for the following types of work:
  - Server process LDUs that are assigned `AVZ_SCHI`.
  - SQL transactions are classified according to WLM classification rules. If a Data Virtualization Manager server classification rule is added that assigns SQL transactions to `AVZ_SCNM` or `AVZ_SCHI`, the LDU representing the transaction is joined to one of the long-running enclaves that are established for the transaction task. A new enclave is created for this LDU.
- The `AVZ_SCTX` service class is used for SQL transactions that are not otherwise classified. A new enclave is created for each LDU assigned to `AVZ_SCTX`.

About this task

To modify the service class definition:

Procedure

1. Select the **Service Classes** option from the **WLM Definition** panel. Press Enter.

   The system displays the **Service Class Selection List** panel.
2. Select a definition in the service class selection list. Select Enter.

   The system displays the following panel that shows the default definition for the AVZ_SCNM service class.

   The description contains the following information:

   - The service class name can be modified by using the `WLMHISERVICECLASS` parameter, which is located in the `AVZSIN00` configuration member.

     Note: Do not change this name unless you are told to do so by Technical Support.
   - The `ZIIPCLASS=AVZ` in the description field is used to construct the names of report classes that have CPU offload criteria that are specified in their descriptions. If the `ZIIPCLASS` keyword is not specified correctly, IBM Data Virtualization Manager for z/OS work that is dispatched as enclave SRBs assigned to this service class is not off loaded to the zIIP.
   - The workload name is for reporting purposes only and can be changed to any valid workload name.
   - The service class goal is a single period with an execution velocity goal. The percentage and importance can be changed, but set them at a level appropriate to a mission-critical server.

Viewing subsystem and classification rules

View the Data Virtualization Manager server classification rules.

About this task

The following classification rules are required:

- The subsystem type must be AVZ.
- A rule classifying transaction `AVZ_TNHI` to service class `AVZ_SCHI`.
- A rule classifying transaction `AVZ_TNNM` to service class `AVZ_SCNM`.
- A rule classifying transaction `AVZ_TNTX` to service class `AVZ_SCTX`.

Procedure

1. Select the **Classification Rules** option from the **WLM Definition** panel (see "Using the WLM Administration Tool").
2. Select **AVZ** from the list of rules in the classification rules selection.
3. Press **ENTER**. The system displays the **Modify Rules for the Subsystem Type** panel that shows the default definition for the AVZ classification rules.

Results

Do not change the classification rules. They are used internally by the Data Virtualization Manager server. Classification rules for SQL, Streams, and Services can be added to these rules.

Modifying a report class definition

Before you begin

The following report classes are required by the Data Virtualization Manager server:

- AVZ_RCP1
- AVZ_RCP2
- AVZ_RCP3

About this task

To modify a report class definition:

Procedure

1. Select the **Report Classes** option from the **WLM Definition** panel. Press Enter.

   The system displays the **Report Class Selection List** panel.
2. Select a report class name from the list of report classes. Press Enter.

   The system displays the following panel that shows the default definition for the report class.

   The panel shows the following information:

   - The report class definition is used to provide first period CPU offload information for service classes. The report class is not used in the classification rules.
   - The report class name can be modified by using the `WLMP1REPORTLASS` parameter, which is located in the `AVZSIN00` configuration member.

     Note: Do not change this name unless you are told to do so by IBM Software Support.

     The format of the report class name is:

     xxx`_RCP1`

     where *xxx* is a ZIIPCLASS=*xxx* specification on a service class description and `_RCP1` is fixed and must not be changed.
   - The D*nnnn* in the description field is the first period duration in service units for CPU offloading. The *nnnn* value can be adjusted by the user.
   - The *P100* in the description field is the percentage of time in the first period that WLM attempts to offload enclave SRBs in the associated service class to the zIIP.

WLM classification rules

WLM classification rules apply to the SQL solution.

Note: Before defining classification rules, make sure that WLM is installed and set up correctly.

SQL

The Data Virtualization Manager server establishes a unique enclave for each transaction. WLM classification rules can assign this enclave to a service class with velocity or response goals and one or more periods.

WLM populates the enclave definition with the following information:

- Client User ID. WLM uses the client user ID to find a classification rule match. The client user ID is mapped to the WLM qualifier type `UI`.
- DB2 Plan Name. WLM uses the DB2 plan name to find a classification rule match. The DB2 plan name is mapped to the WLM qualifier type `PN`.
- DB2 Subsystem Name. WLM uses the DB2 subsystem ID to find a classification rule match. The DB2 subsystem name is mapped to the WLM qualifier type `SPM`.
- Transaction Name. WLM uses the transaction name to find a classification rule match, depending on the following transaction name values. The transaction name is mapped to the WLM qualifier type `TN`.
  - APPLNAME: (Default) The application name that is specified in the client data source is used as the transaction name.
  - MODNAME: The name of the application by using the Data Driver is used as the transaction name.
  - INTNAME: The application executable internal name is used as the transaction name.

Using WLM classifications

You can allow WLM to use their existing service and report classes instead of using the hard-coded IBM Data Virtualization Manager for z/OS definitions.

Procedure

1. Set the following parameters that are located in the AVZSIN00 configuration file. Set the values of the WLMUSERID and WLMTRANNAME parameters to names already in your policy so that IBM Data Virtualization Manager for z/OS is correctly classified.

```
if 1 = 1 then
  do
    “MODIFY PARM NAME(WLMUSERID) VALUE(AVZS)”
    “MODIFY PARM NAME(WLMTRANNAME) VALUE(APPLNAME)”
```

2. If your Data Virtualization Manager configuration member, AVZSIN00, does not match your existing WLM definitions, add the following parameter to your AVZSIN00 member, and keep the default value NO:

```
if 1 = 1 then
  do
    “MODIFY PARM NAME(WLMFORCEPOLICY) VALUE(NO)”
```

Note: If you set WLMFORCEPOLICY to NO, and the service class and report class descriptions are not correct, the zIIP offload criteria is unavailable and the default value of 100% is used for all IBM Data Virtualization Manager for z/OS enclaves. The service and report classes to which reference is made are those set (or defaulted) in the Data Virtualization Manager configuration member, AVZSIN00, for the WLMPnREPORTCLASS and WLM*SERVICECLASS parameters.

Activating the WLM service policy

About this task

⚠ **Warning:** If you change a required element to an invalid value or remove a required definition and activate a service policy, all active servers that require that definition are shut down.

Procedure

1. After you edit the service definition, select **Utilities** from the **WLM Definition** panel (see “Using the WLM Administration Tool”).
2. From the **Utilities** menu, select the **Install Definition** option to save the updated service definition.
3. Use the **Activate Service Policy** option to activate a service policy.

Verifying WLM classification

Procedure

1. Make sure the following started task parameter is added to the AVZSIN00 configuration member:

```
"MODIFY PARM NAME(TRACEWLMCALLS)   VALUE(YES)"
```

This activates tracing for Data Virtualization Manager server calls made to the WLM APIs for transaction management.

2. Connect with your application, and run a transaction.
3. Go to the **Data Virtualization Manager server Primary Option** menu, and select the **Trace Browse** option. Press Enter.
4. The system displays a panel that shows the trace (trace lines are wrapped for the purposes of easier viewing).

The panel shows an ODBC connection that is created from Data Virtualization Manager studio to a Data Virtualization Manager server, and an update that is sent to a DB2 table by using this connection. The AVZSIN00 member contains the following command:

```
"MODIFY PARM NAME(WLMCLASSTRAN)VALUE(YES)"
```

The following classification rule was added to the default rules installed by the Data Virtualization Manager server for subsystem AVZ.

```
____ 1 TN * ___ SDH_SCTX ________
```

The Trace Browse shows the following WLM operations that occurred:

- WLM enclave join executed. The LDU for the new connection thread is joined to the long running AVZ_SCHI enclave to initialize the thread.
- WLM classify work executed. The LDU for the new connection thread is classified to the AVZ_SCTX service class.
- WLM enclave create executed. An enclave is created using the AVZ_SCTX service class for the new connection thread.
- WLM offload CPU time executed. This shows the call to WLM with the criteria for offloading this enclaves SRB work to the zIIP. The durations and percentages for the offloading are obtained from the AVZ_RPCn report class definitions.
- WLM enclave leave executed. The LDU leaves the AVZ_SCHI enclave that it joined to initialize the thread.
- WLM enclave join executed. The LDU is joined to the AVZ_SCTX enclave that was created for it in a preceding step. This is where the actual transaction work is done. In this case, an update is made to the DB2 table USERID.STAFF.
- WLM enclave leave executed. Processing of the DB2 update is complete, and the LDU leaves the AVZ_SCTX enclave.
- WLM enclave join executed. The LDU rejoins the AVZ_SCHI enclave for thread termination.
- WLM enclave delete executed. The AVZ_SCTX enclave is deleted.

WLM Health Reporting

Data Virtualization Manager server reports to WLM on its relative "health" by issuing the IWM4HLTH macro with a health indicator between 0% and 100%. Data Virtualization Manager server starts with a health indicator of 100%. This reporting is enabled by using the WLMHEALTHREPORT parameter, which by default is set to YES.

Periodically, Data Virtualization Manager server examines indicators and adjusts its health percentage. If failures, such as ACI timeouts and ACI abends, occur, the health percentage is adjusted down. The higher the failure rate, the larger the adjustment. If no failures occur, the health percentage is adjusted up. To set the interval for this parameter, use the WLMHEALTHINTERVAL.

Configure WLM health reporting by using the following parameters in the AVZSIN00 member.

Parameter	Description	Valid values
CONCURRENTMX	The maximum number of concurrent sessions, which may be open with the server. This limit is enforced such that new connection requests are rejected if the total number of active sessions would exceed this limit. Setting this limit to zero causes all new connections to be rejected, while allowing in-flight sessions to remain active.	2000 (default)
WLMHEALTHINTERVAL	Controls how often health statistics are reported to WLM. Interval is in seconds.	60 (default)
WLMHEALTHREPORT (*optional*)	Controls whether the Data Virtualization Manager server reports its health percentage to WLM.	**YES** (default) Data Virtualization Manager server uses the current rates of ACI timeouts and ABENDS to compute a change in the health percentage reported to WLM. **NO**
WLMMAXHEALTH (*optional*)	Controls the current health value reported to WLM.	0 – 100 (default value is 100)

You can examine the current level of health by looking at the value for WLMHEALTH by selecting **AVZ Admin** > **AVZ Parms** > **PRODWLM** from the **IBM Data Virtualization Manager for z/OS Server — Primary Option** menu.

You can change the current health value by using the WLMMAXHEALTH parameter.

This parameter allows the CONCURRENTMX parameter to work with the SHAREPORTWLM parameter. Setting CONCURRENTMX parameter to zero forces WLMMAXHEALTH to zero. Setting CONCURRENTMX from zero to nonzero forces WLMMAXHEALTH to 100.

Server load balancing

With load balancing, inbound connections are automatically directed to the server instance that has the most available resources. To determine which instance handles a request takes into account the number of connections currently being handled by each instance and the availability of virtual storage (above and below the 16-MB line).

Load balancing is transparent to the client application. An application uses a port number to connect to an instance, and the instance determines whether it or another instance should handle the session. If another instance is a better choice, the session is transferred.

The client application can be configured with the port numbers of more than one member of the group. This configuration improves reliability by providing a fallback if the copy of the product that uses the base port number is not available. Load balancing increases the number of concurrent connections that the server can handle. As a practical matter, this feature supports far more connections by using RPCs to be concurrently handled. This feature is a key point, because connections by using RPCs exhaust the virtual storage resources of a server instance much faster than DB2 connections.

Sequence of events

Load balancing enables IBM Data Virtualization Manager for z/OS to:

1. Find the address space of the first server.
2. Verify whether sufficient virtual storage exists. If enough storage is available, the server is marked as a candidate.
3. Check the number of active connections.
4. Repeat steps 1 - 3 until all the server candidates are identified.

From the available candidates, the server that has the least number of active connections is selected. If a candidate is not found, the connection is rejected with an inadequate host resources error message.

The group concept

Load balancing is based on the concept of a group. All copies of the server on one system with the same group name are automatically members of the same group. A copy of the product can be a member of only one group at a time or it can be configured to be a stand-alone server and not as a member of a group. All address spaces in a group must reside on the same z/OS image.

The group name can be changed at any time, which means copies of the server can join groups or leave groups as needed.

Using a group director

You can also define a server as a group director. The load balancing group director does no work except route connections to other Servers. A group director passes connections to the best candidate server in the group, giving the user the option to not run application work in a server that accepts inbound connections. When using a group director, everyone connects to the group director, which then routes all of the connections to other Data Virtualization Manager servers.

If you have one Data Virtualization Manager server act as a group director for load balancing, this server would not perform application work, but would only route requests to other servers based on current loads. The server that is performing as a group director has the greatest stability. This method ensures that a connection is made even if some of the application servers are down. This method gives you the highest availability.

If you do not use a group director, the same Data Virtualization Manager server that is routing requests also does application work. If that server fails, all subsequent connection requests fail.

Enabling load balancing for a group director

Procedure

1. Use the `MODIFY PARM` command to set the following parameters that are located in the Data Virtualization Manager configuration member, `AVZSIN00`:

   ```
   if 1 = 1 then
     do
       “MODIFY PARM NAME(GROUPNAME) VALUE(NULL)”
       “MODIFY PARM NAME(GROUPDIRECTOR) VALUE(YES)”
   ```

   The following table lists the parameters for configuring load balancing:

Parameter	Description	Valid values
GROUPNAME	Controls the group that the current instance belongs to. All instance that belongs to the same group (that is, have the same GROUPNAME) automatically load balance among each other. If this value is not set, then the current instance does not belong to a group.	Null
GROUPDIRECTOR	Indicates that a member of the group takes the role of director.	**YES** **NO** Default value is NO.

2. Ensure that everyone connects to the instance that is the group director.
3. Add extra Data Virtualization Manager server subsystems, and set the maximum number of connections in each.

Enabling load balancing for CICS/TS

Procedure

Use the `MODIFY PARM` command to set the following parameters that are located in the Data Virtualization Manager configuration member, `AVZSIN00`:

```
if 1 = 1 then
  do
    "MODIFY PARM NAME(CICSLOADBALANCE) VALUE(YES)"
    "MODIFY PARM NAME(GROUPDIRECTOR) VALUE(YES)"
```

The following table lists the parameters for configuring load balancing for CICS/TS:

Parameter	Description	Valid values
CICSLOADBALANCE	Specifies whether to use the CICS transaction queue depth to decide about load balancing.	**YES** **NO** Default value is NO.
GROUPDIRECTOR	Indicates that a member of the group takes the role of director. The director only accepts inbound connections and pass them to a member of the group that is determined to be the most acceptable in terms of load and resource availability. The group director does not support an application execution environment. This configuration provides a more robust load balancing group.	**YES** **NO** Default value is NO.

Enabling load balancing for Services

A load balancing port is added to IBM Data Virtualization Manager for z/OS Services classes services. To prevent disruptions with existing services and the Data Virtualization Manager studio, the current WSOEPORT remains non-load balanced.

Procedure

Use the **MODIFY PARM** command to set the following parameters that are located in the Data Virtualization Manager configuration member, AVZSIN00:

```
if 1 = 1 then
  do
    "MODIFY PARM NAME(WSOEBALANCEDPORT) VALUE(0)"
    "MODIFY PARM NAME(ZSRVGROUPNAME) VALUE(XXXXXXXX)"
    "MODIFY PARM NAME(ZSRVGROUPDIRECTOR) VALUE(YES)"
```

The following table lists the parameters for configuring load balancing for Services:

Parameter	Description	Valid values
WSOEBALANCEDPORT	Specifies the port number on which to listen for requests.	0
ZSRVGROUPNAME	Controls which Services group, if any, the current copy of the product should belong to. The product uses groups for load balancing across multiple copies (separate subsystems) of the product. All copies of the product that belong to the same group (that is, have the same GROUPNAME) automatically load balance between each other. If this parameter is not set, then the current copy of the product does not belong to any Services group.	NULL
ZSRVGROUPDIRECTOR	Indicates that a member of the Services group takes the role of director. The director accepts only inbound connections and passes them to a member of the group, which is determined to be the most acceptable in terms of load and resource availability. The group director does not support an application execution environment. This provides a more robust load balancing group.	**YES** **NO** Default value is NO.

To enable load balancing without changing the client, users must use the current WSOEPORT as the WSOEBALANCEDPORT and use a brand new port as WSOEPORT.

All group members must share Virtualization Facility libraries (or at least exact copies) and all Virtualization Facility caches must be refreshed after any change. No automated mechanism is in place to synchronize the Virtualization Facility caches. Also, the connection names that are defined in the AVZSIN00 configuration member for all the group members must match the names that are defined in the CICS Target Systems. Each connection has a unique netname that identifies the CICS definition. If this is done, the servers can share Virtualization Facility libraries.

CICS failover

Use CICS failover to set up an alternate CICS ID for each CICS connection, so that if access to a primary CICS connection fails, a hot failover is performed to the alternate CICS region.

CICS failover is on a per Data Virtualization Manager server basis. That is, when you define a CICS connection in your Data Virtualization Manager server, you can specify a primary CICS region and a failover CICS region.

When the primary CICS region is unavailable, the Data Virtualization Manager server automatically routes new and in-flight CICS calls to the failover CICS region. From then on, all transactions are routed to the failover CICS region even after the primary CICS region becomes available again. It is up to the user to decide when to switch back to the primary CICS again.

To switch back to the primary CICS region without disruption, you can log on to the failover CICS, and set the connection that is used by the Data Virtualization Manager server to “Out-Of-Service.” This causes the Data Virtualization Manager server to begin routing transactions back to the primary CICS. Once that takes place, you can put the connection in the failover CICS back “In-Service,” so it can start handling failover support again

This failover support works for both XA and non-XA clients. The only time that in-flight transactions cannot be routed is when an XA data source is used, and your CICS transaction is not a one-for-one (one distributed program link (DPL) request per one unit of work (UOW)). That is, if you have a CICS UOW that consists of multiple DPL requests, and one of the DPL requests fails, this CICS transaction is backed out. This situation is not common because CICS transactions are one-for-one.

Enabling CICS failover

Procedure

Add the following parameter to the `DEFINE CONNECTION` statement in the `AVZSIN00` configuration member:

```
ALTAPPLID(xxxxxxxx)
```

Where *xxxxxxxx* is the application ID (`APPLID`) of the alternate CICS connection. The `ALTAPPLID` is used the connection to the primary CICS region fails.

Note: The `ALTAPPLID` must have the same named connection definitions and application definition that are in the target CICS `APPLID`.

Block fetch

Block fetch pre-extracts rows and sends them in blocks to the requesting node. this process improves the performance of most queries by minimizing network traffic and by using data that is already on the node to accommodate subsequent queries.

Data Virtualization Manager server only uses block fetch with read-only queries. This type of query occurs in the following situations:

- The SELECT statement has a FOR FETCH ONLY clause.
- The SELECT statement has an ORDER BY clause.
- The SELECT statement’s first FROM clause contains more than one table (or view).
- The SELECT statement has the UNION or UNION ALL operator.
- The SELECT statement has the DISTINCT keyword in the first SELECT clause.
- The SELECT statement has a column function in the first SELECT clause.
- The SELECT statement has a HAVING clause in the outside SELECT statement.

- The SELECT statement has a GROUP BY clause in the outside SELECT statement.
- The SELECT statement contains a subquery where the base object of the SELECT statement and the subquery is the same table.

By default, blocks hold 256 KB of data. This number is set by the Data Virtualization Manager server NETWORKBUFFERSIZE parameter. The number of blocks that are used is set by the Data Virtualization Manager server PREFETCH parameter. If Data Virtualization Manager server evaluates a query and determines that it is eligible for block fetch, it begins fetching rows into the prefetch buffers; however, no transmission of data takes place until the first (real) FETCH statement reaches the server.

Note: The maximum number of bytes that is sent for each transmission (for each VTAM SEND) is limited to 32 KB, although Data Virtualization Manager server's internal prefetch buffers can be larger.

Use block fetch to improve the performance of queries that process many rows in a table.

Note: Using block fetch with a query in which no DESCRIBE (or PREPARE INTO) is performed in advance of fetching rows can degrade performance. Data Virtualization Manager server must internally perform a DESCRIBE to determine the types of data that may be returned.

In addition, depending on the type of isolation level that is used, remember the following considerations:

- If the plan is bound with the Repeatable Read (RR) option and block fetch is used, many more pages can be locked for update than without block fetch, especially if the number of rows that are normally extracted by the query is small.
- If the plan is bound with the Cursor Stability (CS) option and block fetch is used, data changes can take place between the time the data is extracted and the time that it is used by the application.

Enabling block fetch

Using block fetch improves performance of certain types of SQL queries by asynchronously pre-extracting rows (on the server node) ahead of the current row. The pre-extracted rows are then sent back to the requesting node in blocks that contain multiple rows of data.

Procedure

To enable block fetch, use the `MODIFY PARM` command to add the following parameter to the `AVZSIN00` configuration member:

```
"MODIFY PARM NAME(PREFETCH) VALUE(3 BLOCKS)"
```

Parameter	Description	Valid values
PREFETCH	Controls how many blocks of rows should be fetched from DB2. These blocks of rows are used to build the compressed row buffers that are sent to an ODBC application from the server. This value should only be changed if the buffers that are being transmitted from the server to an ODBC client application are not full. **Note:** This parameter value should be changed only when recommended by technical support.	3 (default)

Configuring DB2 for z/OS Continuous Block Fetch

You can configure support for DB2 for z/OS Continuous Block Fetch (CBF) using DRDA for high performance.

About this task

This task applies only to IBM DB2 for z/OS using DRDA.

Procedure

1. Configure the AVZSIN00 member.
   a) Set the DRDA configuration for DB2.
   b) In the DRDA Define, the default for the QRBLKSZ parameter is set to 128K. Modify QRBLKSZ if a larger block if needed. Recommendation is to keep the default. As an example, to add 512K:

```
"DEFINE DATABASE TYPE(MEMBER)"              ,
           "NAME(DB3A)"                      ,
           "LOCATION(ZOS3DB3A)"              ,
           "DDFSTATUS(ENABLE)"               ,
           "DOMAIN(MYHOST)"         ,
           "PORT(3740)"                      ,
           "CCSID(37)"                       ,
           "QRBLKSZ(524288)                  ,
           "IDLETIME(160)"
```

2. Add the DRDAMAXBLKEXT parameter. Start with value 8:

```
"MODIFY PARM NAME(DRDAMAXBLKEXT)        VALUE(8)"
```

3. In the SQL query, estimate the number of rows in the RESULT SET and use it in the SQL query as follows:
   a) Assuming the SQL query is `SELECT * FROM CBFTABLE`, and there are 50000000 rows.
   b) Append the following to the end: `OPTIMIZE FOR 50000000 ROWS FOR FETCH ONLY`

   For example:

```
SELECT * FROM CBFTABLE OPTIMIZE FOR 50000000 ROWS FOR FETCH ONLY
```

4. To verify the functionality, turn on the following `TRACE BROWSE` parameter:
   a) `TRACE DRDA CODEPOINT READ/WRITE/FLOW YES`
   b) `TRACE DRDA CODEPOINT WRITE BUFFER YES`

   The trace should look like the following example. The number of corresponding “CodePoint(READ)” equates to the value of DRDAMAXBLKEXT set in the AVZSIN00:

```
18:56:43 0301869847                LEN=02A7,CPT=241B,ELEN=00
18:56:43 0301869848 DSNHLI INTERNAL     OPEN-CURSOR - DSNT400I
                                   SQLCODE = 000,  SUCC
18:56:43 0301869849                LEN=0221,CPT=241B,ELEN=00
18:56:43 0301869850 DSNHLI BLOCK FETCH  (41490) - RC 0 REASON
                                   00000000 SQLCODE 0
18:56:44 0301869851                LEN=01A0,CPT=241B,ELEN=00
18:56:44 0301869852                LEN=007C,CPT=241B,ELEN=00
```

MapReduce

This section provides information on MapReduce features for performance enhancement.

You should also refer to the *IBM Data Virtualization Manager for z/OS User's Guide* for additional information on using MapReduce features.

Virtual Parallel Data

Virtual Parallel Data (VPD) allows you to group multiple simultaneous requests against the same data source and run them in parallel, while doing the input and output (I/O) only once. VPD also allows single or multiple requests to run with asymmetrical parallelism, separately tuning the number of I/O threads and the number of client or SQL engine threads.

To use this feature you must provide a VPD group name when submitting request(s). All requests submitted to the same Data Virtualization Manager server with the same group name within a time period will be placed into a VPD group. One or more I/O threads will be started to read the data source and write it to a wrapping buffer. Group members will share the data in the buffer(s), without having to read the data source directly.

A group is created when the first member request arrives. The group is closed either when all members (and all their parallel MRC threads) have joined, or when a timeout has expired. The I/O threads are started as soon as the group is created, and data begins to flow to the buffer. If the buffer fills before the group is closed, the I/O thread(s) will wait. Once the group is closed and active members begin consuming data, the buffer space is reclaimed and I/O continues.

VPD supports MapReduce Client (MRC), and group members can use different levels of MRC parallelism. For example, a single VPD group might have six members, three members using 5 MRC threads, and the other three using 9 MRC threads. The group will consist of six members and 42 client threads. The number of I/O threads is determined separately. VPD supports a group of a single member, thus supporting asymmetrical parallelism for single requests when using MRC.

VPD is currently supported for the following data sources:

- Adabas files
- Physical sequential data sets on disk, tape, or virtual tape
- Log streams
- VSAM KSDS, RRDS, and ESDS files
- IAM files
- zFS/HFS files

Configuring Virtual Parallel Data
To configure Virtual Parallel Data, specify a group name and appropriate parameters.

Procedure

1. Configure the following parameters in the AVZSIN00 member:

```
/*------------------------------------------------------------------*/
/* Enable Virtual Parallel Data for asymmetrical parallelism      */
/*------------------------------------------------------------------*/
if DoThis then
  do
  "MODIFY PARM NAME(VPDGROUPTIMEOUT) VALUE(60)"
  "MODIFY PARM NAME(VPDBUFFERSIZE) VALUE(40)"
  "MODIFY PARM NAME(VPDTRACEDB) VALUE(NO)"
```

The following table lists the VPD parameters:

Parameter	Description	Valid values
VPDBUFFERSIZE	Specifies the default buffer size, in megabytes above the bar, for a Virtual Parallel Data buffer.	Numeric value in megabytes. Default is 40.
VPDGROUPTIMEOUT	Specifies the maximum time, in seconds, from the time a group is formed until it is closed. Default: 60 seconds	Numeric value in seconds. Default is 60.

Parameter	Description	Valid values
VPDTRACEDB	Controls whether Virtual Parallel Data processing will trace debugging messages.	**NO** Do not trace debugging messages (default). **YES** Trace debugging messages.
VPDTRACEREC	Causes Virtual Parallel Data to trace at the record level. (*Optional*) **Note:** Setting this to YES will produce a large amount of trace output.	**NO** Do not trace record level messages (default). **YES** Trace record level messages.

2. Supply the group name.
3. Optional: Specify the number of members in the group. Although optional, this parameter is recommended.

   When this parameter is provided, the group is closed as soon as all members have joined. If the number is not provided, the group is not closed until the timeout expires. There is no default.
4. Optional: Specify a timeout value for the group formation.

   When the first group member request arrives at the server, the timer is started. If the group remains open when the request expires, it is closed. Any members/threads arriving after the timeout will be placed in a new group. The default is 60 seconds, and can be overridden in the AVZSIN00 file.
5. Optional: Specify the number of I/O threads to use when reading the data source. If this value is not provided, the number of threads is determined as follows:
   a) If the data source is a tape data set and the number of volumes can be determined, the same number of I/O threads will be started.
   b) Otherwise, if a Map Reduce thread count is provided in the data map, that number is used.
   c) Otherwise, if a value is configured for `ACIMAPREDUCETASKS` in the AVZSIN00 configuration member, that number is used.
   d) Otherwise, a single I/O thread will be started.

Innovation Access Method (IAM)

Innovation Access Method (IAM) is a VSAM optimization product distributed by Innovation Data Processing. Enable MapReduce for IAM by setting the `MAPREDUCEIAMKEYMOD` parameter to YES.

MapReduce is implemented by analyzing the file to be retrieved and dividing it up into parts for simultaneous parallel retrieval. For VSAM, this is done by referencing information kept by VSAM about a file. This is supported for key-sequenced data sets (KSDS), entry-sequenced data sets (ESDS), and relative record data set (RRDS) VSAM files. For sequential files, this is done by analyzing information about the extents and volumes of the file. However, for IAM a different approach must be taken because there is no information about the internal structure of an IAM file.

To implement MapReduce for IAM, contact Innovation Data Processing and request module IAMRKTEX. This module will perform the analysis of the internal structure of the IAM file and allow implementation of MapReduce technology. This module will be provided free of charge on request to Innovation Data Processing.

Configuring MapReduce for IAM

Enable MapReduce for IAM by configuring the Data Virtualization Manager server.

Before you begin

The Data Virtualization Manager server must already be installed.

About this task

To enable MapReduce for IAM, you must configure the server configuration file.

Procedure

1. Locate the Data Virtualization Manager configuration member. The server initialization member is shipped in data set member *hlq*.SAVZEXEC(AVZSIN00) and may have been copied to a new data set for customization in the step "Copying target libraries" in the *Installation and Customization Guide*.
2. Locate the parameter `MAPREDUCEIAMKEYMOD`.
3. Use the **`MODIFY PARM`** command to change the MAPREDUCEIAMKEYMOD parameter value, as follows:

   ```
   "MODIFY PARM NAME(MAPREDUCEIAMKEYMOD) VALUE(YES)"
   ```

Metadata repository

The metadata repository for MapReduce stores statistics about virtual tables that are used to enhance performance in conjunction with MapReduce and parallelism. This support applies to DRDA and IMS data sources, including those accessed via the IBM Federated Server (such as Terradata and Sybase), as well as data sources accessed via direct DRDA support (DB2 LUW and Oracle) provided by the Data Virtualization Manager server. The gathered metadata persists across server restarts.

Populating the metadata repository

You can periodically run the **DRDARange** command to gather metadata repository information about the backend virtual tables. In case of an IMS data source, you can use IMS Direct to gather the metadata repository information.

About this task

You can run the metadata repository command for DRDA either using the ISPF panels or a batch job.

Note: When using MapReduce support, **DRDARange** is required for a relational database management system (RDBMS).

The following restrictions and considerations apply when using this feature:

- Current support does not contain any optimizer enhancements for processing complex queries or joins other than what may be used to enhance MapReduce.
- If a table does not contain enough rows to properly calculate a DRDA Range, then the following error is also returned for this condition:

  ```
  Table <schema>.<table_ name> not eligible for range processing
  ```

  An additional error message can be found in the tracebrowse for this error. For example:

  ```
  22:10:53 Row count 14 too small for range processing
  22:10:53 SELECT DRDARANGE('virtual_table.DBLIDX') FOR FETCH ONLY - SQLCODE 0
  22:10:53 SQL ENGINE HPO OPEN-CURSOR - SQLCODE 0
  22:10:53 SQL ENGINE HPO FETCH - SQLCODE 100
  ```

Procedure

Run the appropriate command as follows:

- Using the ISPF panels:
  - For DRDA data sources, use the SELECT statement at the virtual table level.

    ```
    SELECT DRDARANGE('<TABLE NAME>',MAX_SCAN,'OPTION1','OPTION2',...);
    ```

    Note: It is recommended to use option PARTONLY for partitioned tables. Using this option will force the use of partition boundaries when determining parallelism.

– Using a batch job, which you can use to schedule the commands to refresh the statistics on a specified schedule. A sample job is provided in *hlq*.SAVZCNTL(AVZRANGE). Instructions for required edits to the job are provided in the member.

```
//RANGE  EXEC  PGM=AVZXMAPD,PARM='SSID=AVZS,,MXR=30000000'
//STEPLIB  DD DISP=SHR,DSN=loadlibrary
//RPT      DD SYSOUT=*
//FMT      DD SYSOUT=*,DCB=LRECL=4096
//OUT      DD SYSOUT=*
//IN       DD *
SELECT DRDARANGE('<TABLE NAME>',MAX_SCAN,'OPTION1','OPTION2',...);
```

Chapter 6. Configuring rules and events

Using a rule, you can configure an automatic response to an event. For example, you can allow a critical application to download data any time, and allow a non-critical application to download data only during specific hours.

For example, to restrict the number of times that a user ID can log on to the server, create a LOGON rule to limit the user ID to three logons a day and to take a specific action if the user ID tries to log on more than three times.

Events

You can create rules for the following types of events:

- Authorization (ATH) events that occur when the Data Virtualization Manager configuration performs authorization processing for a controlled resource.
- Command (CMD) events that occur when the Data Virtualization Manager configuration receives a command from a z/OS® console.
- Exception (EXC) events that occur when tasks exceed limits or fail. These events are generated only when the SEFGLVENTS parameter is set to allow them.
- Global variable events (GLV) that occur when the value of a global variable is changed.
- Publish (PUB) events occur in response to INSERT, UPDATE, and DELETE.
- RPC events occur in response to remote procedure calls.
- SQL events occur before a SQL statement is run.
- Time-of-day (TOD) events occur at specific times.
- Virtual tables (VTB) rules allow you to have a single virtual table that can use to represent many data sets of the same structure.

For each event, you can create one or more rules. Within each rule, you specify an action to take in response to the event. For example, you might create two rules for the LOGON event. In one rule, you specify that if an ID attempts to log on more than three times within a 24-hour period, subsequent logon requests are rejected. In another rule, you might specify that all logs on attempts from a specific ID are rejected.

Rules and rule sets

A rule can have the following parts:

- Criterion
- Header statement
- One or more process sections
- Return values
- Variables

Automatic limits

A rule can include customizable limits that control many aspects of your configuration including queries, connections, and sessions.

Rules are configured in the Data Virtualization Manager configuration member that is shipped in data set member *hlq*.AVZSIN00.

You can view rules by selecting **E (AVZ Admin.)** > **2 (AVZ Parms)** from the Primary Option Menu. To modify a rule, locate the parameter, change its value, and press **Enter**. This modifies the parameter for the existing Data Virtualization Manager session. To make the change permanent, modify the parameter in the AVZSIN00 configuration member.

During installation, a default value is specified for each of the following limits.

Overall per session CPU limit
When this limit is reached, the session is automatically terminated. The security product or a product parameter can provide the limit.

Per Db2® connection CPU limit
When this limit is reached, the current Db2 connection is automatically terminated, and all associated Db2 resources are released.

Per SQL query CPU limit
When this limit is reached, the current SQL query is automatically terminated, and all associated Db2 resources are released.

Inactivity time-out
This limit automatically terminates the session of any user that is inactive for the specified period. Use this limit to minimize security exposures and release resources that are held by inactive users.

Maximum timer-on limit
This limit prevents the execution of any SQL statement that exceeds a specified value. The limit prevents excessive resource utilization.

Maximum rows limit
This limit restricts the number of rows that a query returns.

RPC maximum rows limit
This limit restricts the number of rows that an RPC can generate. This limit is set as a host parameter and is enforced on the host (Data Virtualization Manager server).

Dropped connection detection
This mechanism detects clients that failed or are no longer connected to the network. When a dropped connection is detected, the host session is terminated, and all resources are released.

Lock control facility
This mechanism detects clients that are holding a Db2 lock (share, update, or exclusive) for an excessive period. When the limit is reached, the session is terminated, and the lock is released.

Dynamic SQL control facility
This mechanism allows dynamic SQL to be rejected on the host. Use this mechanism to enforce the use of static SQL.

Maximum concurrent users
This limit controls the maximum number of concurrent users and is enforced on the host.

Variables for rules

When you create a rule, you can use dynamic variables, global variables, temporary variables, and event-specific variables. These variables are used in REXX programming.

Dynamic variables

Dynamic variables are created when the process section of a rule references or sets the value of a simple or compound variable. Dynamic variables exist only while a rule runs and are freed when the REXX environment is deleted. Dynamic variables cannot be accessed by non-REXX procedures and functions. The following code fragment shows two simple variables, I and COUNT, and one compound variable, stemvar.I:

```
do I = 1 to COUNT
stemvar.I = "InitValue"
end
```

Global variables

Global variables have one of the following stem values:

- GLOBAL
- GLOBAL*n*, where *n* is an integer 1 - 9

Global variables can be created, modified, or managed by selecting option **E** (Rules Mgmt.) from the Data Virtualization Manager server - Primary Option Menu and then selecting **1** (Global Variables). To create a new global variable, enter S *variable_name* and press Enter.

Global variables are stored in the global variable checkpoint data set. When a global variable is referenced, the value of the variable is retrieved from the checkpoint data set. The value of a global variable persists across restarts of the product and is shared by all rules. If the **SEFGLVEVENTS** parameter is set to YES in the Data Virtualization Manager configuration member AVZSIN00, you can create a rule to intercept the change and perform additional processing.

Temporary variables

Temporary variables, which begin with the stem value GLVEVENT, exist only during an event and are deleted when the event is over. Temporary variables are used by high-level language (HLL) routines that create and interrogate these types of variables. To create or access a temporary variable, use the SDBVALUE API function. A rule can reference a temporary variable by name.

Event variables

When an event occurs, event variables are created. These variables pass information about the event to the rules for the event. For example, ATH.AUPWDBSS is an event variable for the LOGON event. The value of the ATH.AUPWDBSS variable is the Db2 subsystem name that the connection string provides. You can use this variable in a rule that restricts logons to a specific Db2 subsystem.

Most event variables are read-only; however, some can be modified. Changes to modifiable event variables are cumulative. The first rule that runs uses the original value of the variable. Each rule that later runs uses the value that the previous rule modified. Even if a rule modifies the value of a variable, all rules that are eligible to run still run.

Authorization (ATH) events

This section describes the types of authorization (ATH) events.

All authorization events

This event occurs when an authorization request is made. A rule for this event can reject, accept, or modify the request.

Return values

When an ATH event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.

Return value	Description
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

Variables

Values for these variables are set only when an ATH rule processes an ATH event.

Criterion	Variable name	Contents	Data type
ALL (all variables)	ATH.OPAU13WA	The WAITS flag is on if the wait state is allowed and is off if wait state is not allowed. If the wait state is not allowed, actions that cause the task to enter a wait state are not allowed.	Character, read only
ALL	ATH.OPAUACSR	The type of access that is being requested. The following are valid values for the access type, except for LOGON requests: • ADD • CONTROL • DISPLAY • DEFINE • EXECUTE • INFO • LIST • KILL • MODIFY • READ • SHOW • SET • WRITE	Character, read only
ALL	ATH.OPAUERMG	A REXX program can specify the error message to send to the client.	Character, read-write

Criterion	Variable name	Contents	Data type
ALL	ATH.OPAURQRC	The request return code. The following are valid values: • 00: Request allowed • 04: Request must be modified • 08: Request failed • 12: Request abended • 16: Product address space is unavailable	Character, read-only
ALL	ATH.OPAURQSR	The type of request that is being processed. The following are valid values: • CICSCONNECTIONS: CICS® connections • CONTROLBLOCKS: Product control blocks • DATABASES: Product databases • DATAMAP: Data map definitions • FILE: Shared server QSAM/BPAM data sets • GLOBALS: Global variables • LINKS: Communication links • LOGON: Password and user validation • PARMS: Product parameters • RPC: Remote procedure call • AVZ: AVZ command • SEF: Event Facility commands • TRACEDATA: Detailed Trace Browse data • TRACEBROWSE: Trace browse • TSO: Time Share Option • USERS: Remote users	

Criterion	Variable name	Contents	Data type
ALL	ATH.OPAUSRID	The search ID, which is created by combining the request type with the access type, for example: • PARMS.SHOW displays a product parameter • SEF.INFO obtains SEF information.	
ALL	ATH.OPAUUSID	The user ID that is being validated (LOGON), the user ID being logged off (LOGOFF), or the user ID for the task that is requesting access to the controlled resource. **Note:** A rule for the LOGON event can change the value of the user ID so that the rule-generated user ID can be used for subsequent validation by the security product. Rules for other authorization events should not attempt to alter the ATH.OPAUUSID variable.	Character, read-only, except as noted
ALL	ATH.USER	The user area is passed to all rules that run in response to the same event.	Read-only

Control block events

This event occurs when a control block is accessed or updated. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an ATH event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.

Return value	Description
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

Variables

CONTROLBLOCK variables are used for events that pertain to accessing or updating a product control block.

Variable name	Contents	Data type
ATH.AUBKCBAD	The address of the control block.	Character, read-only
ATH.AUBKCBAS	The address space (ASID) of the control block.	Numeric, read-only
ATH.AUBKCBLN	The length of the control block.	Numeric, read-only
ATH.AUBKCBNA	The name of the control block.	Character, read-only

Database events

This event occurs when a database is defined, accessed, or updated. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When a database event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

Variables

DATABASE variables are used for events that pertain to defining, accessing, or updating a product database.

Variable name	Contents	Data type
ATH.AUDBHOST	The host name of the database.	Numeric, read-only
ATH.AUDBNAME	The name of the database.	Character, read-only
ATH.AUDBTYPE	The type of the database.	Character, read-only

Global variable events

This event occurs when a global variable is defined, accessed, or updated. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an ATH event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determination to allow or deny access to the requested resource.

Variables

The following variables are available.

Variable name	Contents	Data type
ATH.AUGLDELN	The length of the name of the global variable.	Numeric, read-only
ATH.AUGLDENA	The name of the global variable.	Character, read-only

Variable name	Contents	Data type
ATH.AUGLOPCH	The operation. The following are valid values: • A: Add a global variable. • D: Drop a global variable. • E: Check for the existence of a global variable. • F: Check for the existence of a global variable and obtain (return) the value. • I: Obtain information about a global variable. • L: List information about a global variable. • O: Obtain a global variable. • R: Remove a global variable. • S: Subtree processing. • T: Subtree information processing • U: Update a global variable. • V: Value processing.	Character, read-only
ATH.AUGLRQTY	The type of the access request. The following are valid values: • A: READ access • U: UPDATE access	Character, read-only

IMSLTERM events

This event occurs when the IMSLTERM (IMS logical terminal) authorization event occurs. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an IMSLTERM event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.

Return value	Description
Other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

Variables

The following variable is available. The IMSLTERM variable is used for events that pertain to IMSLTERM.

Descriptive name	Variable name	Contents	Data type
Virtual table name	ATH.AULTNAME	The name of the virtual table.	Character, read-only

Communication link events

This event occurs when a communication link is defined, accessed, or updated. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an communication link event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

Variables

LINKS variables are used for events that pertain to defining, accessing, or updating a communication link.

Variable name	Contents	Data type
ATH.AULIHOST	The host name for the link. This name might be truncated. To avoid the additional processing that is required to resolve the host name, the server does not usually obtain or provide the client host name.	Character, read-only
ATH.AULIIPAD	The TCP/IP address in 4-byte binary form.	Binary, read-only

Variable name	Contents	Data type
ATH.AULILU	The LU 6.2 name that is set only for LU 6.2 links.	Character, read-only
ATH.AULIMODE	The LU 6.2 mode name that is set only for LU 6.2 links.	Character, read-only
ATH.AULITYPE	The link type. The following are valid values: • 6: LU 6.2 link • T: IBM TCP/IP link • I Interlink TCP/IP	Character, read-only

Log off events

This event occurs after the client session to the host is terminated. Therefore, no response data can be sent to the client.

A rule for this event can provide the following responses:

- Write messages to a console or to the Trace Browse. The error message variable (ATH.OPAUERMG) can also be set. This value of this variable displays in the Trace Browse if ATH messages are being traced.
- Write SMF records. The SDBINFO function can be used in addition to the ATH event variables passed to this routine.
- Access and update other resources. For example, a global variable can be modified to show that the current user is no longer connected.

Return values

When an log-off event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

Variables

LOGOFF variables are used for events that pertain to writing messages to a console or Trace Browse, writing SMF records, or accessing and updating other resources.

Descriptive name	Variable name	Contents	Data type
Termination code	ATH.AULGABCD	The termination code, which is a 4-byte hexadecimal string. The value is 0000 if the current thread terminated normally.	Character, read-only
Authorization scheme	ATH.AULGAUSC	The authorization scheme. The following are valid values: • SDBECURE: The user ID was created by using the SDBECURE API. • RA-PROXY: A RUNAUTH (proxy) user ID log off. • BASIC: An HTTP authorization, request header scheme.	Character
Cache	ATH.AULGCAUS	The user ID cache flag. The following are valid values: • 0 (zero): The user ID is logged off. • 1: If the user ID was previously cached and is retained in the cache.	Character, read-write
Connection token	ATH.AULGCNTK	The connection token is an 8-byte hexadecimal string. To identify the terminating task, this value can be passed to the SDBINFO function. This value is only required for test (TSO) versions of the main product address space.	Character, read-only
CPU time	ATH.AULGCPTM	The CPU time that is used by the current task, which is specified in seconds and fractions of a second.	Character, read-only
Elapsed time	ATH.AULGELTM	The elapsed time of the current task, which is specified in seconds and fractions of a second.	Character, read-only
GMT logon time	ATH.AULGLGGM	The GMT logon time, which is provided as a timestamp. The format is *YYYY/MM/DD-HH:MM:SS.NNNNNN.*.	Character, read-only
Local logon time	ATH.AULGLGTM	The local logon time, which is provided as a timestamp. The format is *YYYY/MM/DD-HH:MM:SS.NNNNNN.*.	Character, read-only

Descriptive name	Variable name	Contents	Data type
Uncompressed bytes	ATH.AULGWRTO	The total number of uncompressed bytes. It is provided by using the next field.	Character, read-only
Wait	ATH.APAU13WA	The WAITS flag. The following are valid values: • 0 (zero): WAITS are not allowed. • 1: WAITS are allowed. If WAITS are not allowed, I/O and other services that might cause the task to enter a wait state are not allowed. Some logoff operations occur during end-of-task processing when it is important to monitor the wait-allotted flag to prevent unwanted subtask terminations.	

Log on events

This event occurs when a logon occurs.

A rule for this event can provide the following responses:

- Set or reset all of the execution limits for the current client user ID. The default values are passed to the rule. If the default values are not changed, they are used.
- Set the return value to REJECT, and use the ATH.OPAUERMG variable to send an error message.
- Set the return value to ACCEPT. Using this return value bypasses the password validation that the security product does. Use ACCEPT only if you do not have a security product that is installed and rely on
- Modify the user ID before the security product processes it.

Return values

When an ATH event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.

Return value	Description
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

LOGON variables are used for events that pertain to setting or resetting execution limits for the current client user ID, rejecting the current logon attempt, bypassing password validation, or modifying a user ID before it is processed by RACF/ACF2.

Descriptive name	Variable name	Contents	Data type
Security optimization	ATH.AUPWAEAC	The Security Optimization flag. The following are valid values: • 0 (zero): Security optimization is not active. • 1: Security optimization is active.	Character, read-only
Security optimization cache	ATH.AUPWAERT	The amount of time, in seconds, that the security optimization cache entry is retained for the user.	Character, read-only
Application name	ATH.AUPWAPNA	The name of the application. This value is optionally set by the ODBC application.	Character, read-write
Authentication scheme	ATH.AUPWAUSC	The authentication scheme for the logon. The following are valid values: • SDBECURE: A logon by using the SDBECURE API • RA-PROXY: A RUNAUTH (proxy) user ID logon • BASIC: An HTTP authorization, header user ID logon	Character, read-write
User ID cache	ATH.AUPWCAUS	A user ID cache flag. The following are valid values: • 0 (zero): Suppresses caching for the user ID • 1: If the client user ID/acee (access control element entry) is or could be cached for reuse.	Character, read-only
ODBC connection string	ATH.AUPWCNSR	The ODBC connection string from the client.	Character, read-write
Base CPU time interval	ATH.AUPWCPBA	The base CPU time interval for time slicing.	Character, read-write

Descriptive name	Variable name	Contents	Data type
Error CPU time limit	ATH.AUPWCPER	The error CPU time limit that is checked by the check limits task.	Character, read-write
Failure CPU time limit	ATH.AUPWCPFA	The failure CPU time limit that is checked by the check limits task.	Character, read-write
Execution time interval	ATH.AUPWCPIN	The execution time interval for time slicing.	Character, read-write
CPU time limit	ATH.AUPWCPTM	The CPU time limit that is checked by the ODBC task.	Character, read-write
Plan name	ATH.AUPWDBPN	The plan name. This value is provided in the connection string.	Character, read-write
Db2 subsystem name	ATH.AUPWDBSS	The Db2 subsystem name. This value is provided in the connection string.	Character, read-write
Database user ID	ATH.AUPWDBUS	The database user ID that is used to connect to Db2. When you use CAF, you can switch the user ID, but you cannot switch the user ID with RRSAF unless you are using Enterprise Auditing.	Character, read-write
Task priority	ATH.AUPWDPPR	The z/OS task dispatch priority of the current task, which is a value 0 - 225.	Character, read-write
Enterprise auditing	ATH.AUPWENTL	The enterprise auditing flag. If this flag is set to 1, enterprise auditing requests from the client are accepted. If the flag is set to any other value, requests are ignored.	Character, read-write
Exclusive lock	ATH.AUPWEXFA	The exclusive lock time limit, which is checked by the check limits task.	Character, read-write
Application internal name	ATH.AUPWINNA	The application internal name, if available. This value, which is available only for non-console-mode Windows 32-bit applications, is obtained from the Windows version resources.	Character, read-only

Descriptive name	Variable name	Contents	Data type
New plain-text password	ATH.AUPWLGNW	A new plain-text password, which the application provides. The PROVIDEPASSWORDS parameter controls this variable. If the PROVIDEPASSWORDS is set to YES, the variable is set to a non-blank string. Otherwise, the variable is set to blank characters. The password can only be changed if the PROVIDEPASSWORDS parameter is set to CHANGE.	Character, read-write
Plain-text password	ATH.AUPWLGPW	The plain-text password, which the application provides. The PROVIDEPASSWORDS parameter controls this variable. If the PROVIDEPASSWORDS is set to YES, the variable is set to a non-blank string. Otherwise, the variable is set to blank characters. The password can only be changed if the PROVIDEPASSWORDS parameter is set to CHANGE.	Character, read-write
Network user ID	ATH.AUPWLNID	The network user ID from the client.	Character, read-write
Application module name	ATH.AUPWMDNA	The application module name, if available. This is the name of the application that is using the .NET client.	Character, read-only
Maximum rows generated	ATH.AUPWMXCA	The maximum number of rows that a call RPC can generate before an error is reported to the RPC.	Character, read-write
Maximum rows fetched	ATH.AUPWMXRW	The maximum number of rows that can be fetched before SQL code +100 is simulated.	Character, read-write

Descriptive name	Variable name	Contents	Data type
Maximum timerons	ATH.AUPWMXTM	The maximum timerons limit, which is checked by the client task. A timeron is a unit of measurement used to give a rough relative estimate of the resources, or cost, required by the database server to execute two plans for the same query. The resources calculated in the estimate include weighted CPU and I/O costs.	Character, read-write
Single logon	ATH.AUPWNTLG	The single logon flag from the client. The following are valid values: • 0 (zero): The client did not use a single logon. • 1: The client used a single logon.	Character, read-only
RPC enqueue limit	ATH.AUPWRPEH	The RPC enqueue time limit that the check limits task checks.	Character, read-write
RPC execution limit	ATH.AUPWRPEL	The RPC execution time limit.	Character, read-write
Share lock limit	ATH.AUPWSHFA	The share lock time limit that the check limits task checks.	Character, read-write
Per SQL CPU limit	ATH.AUPWSQFA	The per SQL CPU time limit that the check limits task checks.	Character, read-write
Update lock limit	ATH.AUPWUPFA	The update lock time limit that the check limit task checks.	Character, read-write
User parameter	ATH.AUPWUSPA	The User parameter from the client.	Character, read-write
PassTicket authentication	ATH.AUPWSPT	The PassTicket flag. The following are valid: • 0 (zero): The user is not using a PassTicket for authentication. • 1: The user is using a PassTicket for authentication.	Character, read-write
Error wait time	ATH.AUPWWAER	The error wait time limit that the check limits task checks.	Character, read-write

Descriptive name	Variable name	Contents	Data type
Failure wait time	ATH.AUPWWAFA	The failure wait time limit that is checked by the check limits task.	Character, read-write
Warning wait time	ATH.AUPWWAWN	The warning wait time limit that is checked by the check limits task.	Character, read-write
WAITS flag	ATH.OPAU13WA	The WAITS flag. The following are valid values: • 0 (zero): WAITS are not allowed • 1: WAITS are allowed If WAITS are not allowed, I/O and other services that might cause the task to enter a wait state are not allowed.	Character, read-write
Accept type string	ATH.OPAUACSR	The accept type string.	Character, read-only
Error message	ATH.OPAUERMG	The error message.	Character, read-only
Request type string	ATH.OPAURQSR	The request type string.	Character, read-only
Rule-invocation match string	ATH.OPAUSRID	The rule-invocation match string.	Character, read-only
Client user ID	ATH.OPAUUSID	The client user ID being logged on to the system.	Character, read-only

MQ events

This event occurs when an IBM MQ resource is defined. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an MQ event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

The MQSERIES variable is used for authorization of events that pertain to defining an MQ resource.

Descriptive name	Variable name	Contents	Data type
Queue manager	ATH.AUMQQMGR	The name of the queue manager. This name is set only for actions that are specific to one queue manager. This field is not set when the list of queue managers is being requested by a caller.	Character, read-only

Parameter events

This event occurs when a parameter is updated or accessed. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When a parameter event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

The PARMS variable is used for authorization of events that pertain to accessing or updating a product parameter.

Descriptive name	Variable name	Contents	Data type
Product parameter name	ATH.AUPAPANA	The product parameter name.	Character, read-only

RPC events

This event occurs when an attempt is made to run an RPC. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an RPC event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

RPC variables are used for authorization of events that pertain to execution of an RPC.

Descriptive name	Variable name	Contents	Data type
RPC module name	ATH.AURPNASR	The RPC module name that was extracted from the SQL call statement.	Character, read-only
SQL code REXX variable	ATH.AURPSQCD	The SQL code REXX variable. If RPC execution is rejected, the variable is set to a negative value.	Character, read-only

AVZ events

This event occurs when an attempt is made to run the AVZ command. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an AVZ event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

AVZ variables are used for authorization of events that pertain to execution of an AVZ command.

Descriptive name	Variable name	Contents	Data type
Options string	ATH.AUSDOTSR	The AVZ command Options string, such as 5.2.	Character, read-only
Subsystem name	ATH.AUSDSSNA	The subsystem name.	Character, read-only

SEF events

This event occurs when an attempt is made to run the SEF (event facility) command runs. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When an SEF event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

SEF variables are used for authorization of events that pertain to the running of an SEF command.

Descriptive name	Variable name	Contents	Data type
Subcommand for the SEF ARCHIVE verb	ATH.AUSEARSB	The subcommand for the SEF ARCHIVE verb.	Character, read-only
Current® operation	ATH.AUSEAUOP	A flag that shows if the current operation affects the event procedure rule set. The following are valid values: • 0 (zero): • 1:	Character, read-only
Rule set name	ATH.AUSEAURS	The ATH rule set name.	Character, read-only
Command buffer length	ATH. AUSEBULN	The SEF command buffer length.	Character, read-only
Command buffer	ATH.AUSECMBU	The SEF command buffer.	Character, read-only
z/OS dsname	ATH.AUSEDSNA	The SEF rule set z/OS data set name (dsname for file management commands).	Character, read-only

Descriptive name	Variable name	Contents	Data type
Event procedure name	ATH.AUSERLNA	The SEF command event procedure name (member name for file management commands).	Character, read-only
Command request	ATH.AUSERQTY	The SEF command request type. The following values are valid for rule set commands: • A: Set auto-enable flags • B: Set auto-enable flags and enable them • C: Reset auto-enable flags and disable them • D: Disable rules • E: Enable rules • F: Refresh rules • I: Set dsname index (dsname with STAR) • L: List rule set or rule • R: Archive comand • S: Set or resent subsystem string • T: Test timer rules or another test • U: Show rule • X: Transfer data • Y: Set or reset SYSID string • Z: Reset auto-enable flag	Character, read-only
		The following values are valid for file-management commands: • 3: Open a data set • 4: Close a data set • 5: Refresh a data set • 6: File list • 7: Quiesce a data set • 8: Allocate a data set • 9: Deallocate a ddname	

Descriptive name	Variable name	Contents	Data type
		The following values are valid for TSO server management commands: • F: TSOSRV_LIST • K: TSOSRV_QUEUES • M: TSOSRV_STOP • O: TSOSRV_RESETQ • P: TSOSRV_FREE • Q: TSOSRV_EXECSTATUS	
SEF rule set name	ATH.AUSERSNA	The SEF command rule set name (ddname for file-management commands).	Character, read-only
SEF command verb string	ATH.AUSEVBSR	The SEF command verb string.	Character, read-only

Token events

This event occurs when a token is accessed. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When a token event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

TOKENS variables are used for authorization of events that pertain to the access of an execution token.

Descriptive name	Variable name	Contents	Data type
Host name	ATH.AUTKHONA	The host name field, which contains the host name of the client that created the current token. This field is not set for multiple token fetch requests.	Character, read-only
ID string	ATH.AUTKIDSR	The token ID string, which contains the token ID that is being accessed or deleted. This field is not set for multiple token fetch requests.	Character, read-only
User data	ATH.AUTKUSDA	The user data field, which contains the user data of the token that is being accessed or deleted. This field is not set for multiple token fetch requests.	Character, read-only
User ID	ATH.AUTKUSID	The user ID field, which contains the user ID of the client that created the current token. This field is not set for multiple token fetch requests.	Character, read-only

TSO events

This event occurs when a TSO command runs. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When a TSO event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.

Return value	Description
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

TSO variables are used for authorization of events that pertain to execution of a TSO command.

Descriptive name	Variable name	Contents	Data type
Buffer length	ATH.AUOSBULN	The TSO command buffer length.	Character, read-only
Buffer	ATH.AUOSCMBU	The TSO command buffer.	Character, read-only
Command verb string	ATH.AUOSVBSR	The TSO command verb string.	Character, read-only

User events

This event occurs when information about a remote user is accessed, when a remote user connection is terminated, and when a cancel Db2 thread operation occurs. A rule for this event can accept or reject the request or allow the security product to determine if the request is allowed.

Return values

When a user event ends, the rule sets a return value. The server evaluates the return value and invokes z/OS security routines.

Return value	Description
ACCEPT	Access to the requested resource is allowed, and additional processing by the z/OS security subsystem is not performed.
REJECT	Access to the requested resource is denied, and additional processing by the z/OS security subsystem is not performed. The rule can include the ATH.OPAUERMG variable, which for most authorization requests, returns an error message to the requestor.
Any other value	If another value or no value is returned, the z/OS security subsystem performs validation checking. The security product makes the final determine to allow or deny access to the requested resource.

USERS variables are used for authorization of events that pertain to accessing or killing connections of a remote user.

Descriptive name	Variable name	Contents	Data type
Connection ID	ATH.AUUSCNID	The connection ID, which is set only for stop or cancel operations.	Character, read-only
User name	ATH.AUUSKILL	The name of the user to stop or cancel.	Character, read-only

Descriptive name	Variable name	Contents	Data type
Connection type	ATH.AUUSTYPE	The connection type. The following are valid values: AMDETRT: If a user is requesting information about a specific APPC/MVS conversation information for each task with an active conversation. AMINTSUM: If a user is requesting information about the APPC/MVS summary. DETAIL: If a user is requesting information about user or interval detail data stored in the main product address space. IDDETRT: If a user is requesting information about specific APPC/ IDMS conversation information for each task with an active conversation. REMOTE: If a user request information about all remote connections in the main product address space. REMOTEGRP: If a user is requesting information about TCP/IP host name and port information. RRRMINFO: If a user is requesting information about Resource Recovery Services. SECOPT:If a user is requesting information about security optimization cache entries. SUMMARY: If a user is requesting information about all of the summary interval data stored in the main product address space. TASKS: If a user is requesting information about all tasks that run in the main product address space.	Character, read-only

Descriptive name	Variable name	Contents	Data type
		REMOTE: If a user request information about all remote connections in the main product address space. REMOTEGRP: If a user is requesting information about TCP/IP host name and port information. RRRMINFO: If a user is requesting information about Resource Recovery Services. SECOPT: If a user is requesting information about security optimization cache entries. SUMMARY: If a user is requesting information about all of the summary interval data stored in the main product address space. TASKS: If a user is requesting information about all tasks that run in the main product address space.	

Command (CMD) events

Command events control client/server access to the mainframe.

When the Data Virtualization Manager server receives a command from a z/OS console, a rule is scheduled to run. The console can be a physical console or extended software, such as System Display and Search Facility (SDSF) or CA OPS/MVS Event Management and Automation. The command consists of a command verb, followed by optional operands. The command verb string is matched against enabled CMD rules to find the rule to run.

CMD rules perform the following tasks:

- Examine the command, parse the operands, and perform necessary actions, such as read and set product parameters. This allows parameters to be displayed and changed from the z/OS console.
- Access and update REXX global variables.
- Use REXX SAY statement to communicate with the console that entered the command. All output from the SAY statement is routed to the console that entered the original command. This allows ASO products to communicate with, interrogate the status, and control the Data Virtualization Manager server.

  Note: Because CMD rules can access and update any part of the product, you must control who can create, enable, and disable CMD rules.

All CMD rule processing is done by IBM Data Virtualization Manager for z/OS/REXX. Processing in another programming language is not supported.

Syntax

To trigger a CMD rule, use the z/OS STOP or MODIFY command, or use a z/OS command that specifies the subsystem name. The following commands are valid:

- `MODIFY` *`x`*`VZ`*`y`*`, `*`command text`*

- *x*VZ*y command text*
- *x*VZ*y*, *command text*

where *x*VZ*y* is a specific instance of the Data Virtualization Manager server, which is identified by the subsystem name that was assigned during installation.

When the z/OS STOP command triggers a CMD rule, the rule can control or reject product shutdown. The criterion of the rule must be STOP or a less specific criterion that matches the STOP command. The z/OS STOP (P) command can also trigger a CMD rule that has the matching criterion of STOP.

Header statement

A CMD criterion is a string of 1 - 30 characters. To schedule the rule to run for all commands, use a single * (asterisk) as the criterion. Use a trailing * (asterisk) as a wildcard character.

Use the following format for the header statement:

```
/*CMD criterion
```

Process section

A REXX process section is required.

Return values

The following table lists the return values for CMD rules:

Return value	Action
None supplied	If the rule runs a RETURN command, the Data Virtualization Manager server sends a return code that indicates the successful completion of the rule.
ACCEPT	The command in the rule was successfully completed.
REJECT	The command in the rule was rejected. To specify why the command was rejected, you REXX SAY statements.

The return value for a STOP CMD rule determines how the Data Virtualization Manager server terminates. The following return values are valid:

Return value	Action
None supplied	Termination is allowed to continue.
ACCEPT	Termination is not allowed to continue.
REJECT	Termination is not allowed to continue.

CMD event variables

Values for these variables are set only when a CMD rule processes a CMD event.

Variable	Contents	Data type
CMD.TEXT	Operands that are entered after the command name at the console.	Character, read-only

Variable	Contents	Data type
CMD.VERB	The command name that is entered at the console.	Character, read-only

Exception (EXC) events

An exception event occurs when a task exceeds a specified limit.

The EXC procedure samples that are distributed with the server contain a sample for each of the exception types. Instructions in the samples explain the following information:

- The environment in which the exception is detected.
- The operational controls that affect subsequent processing by the server.
- The valid return values.

The header statement for an EXC rule is /*EXC *criterion*, where *criterion* is one of strings in the following table. A process section is required.

Criterion	Description	Default action
CPULIMIT	A transaction task exceeded its maximum CPU time limit. This exception is detected only when multipart messages are being transmitted and only when a new message segment is being read. A rule for this event can take one or more of the following actions: • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread. This action prevents the exception from occurring again. Use the return value REJECT to terminate the ODBC connection, and use the EXC.OPERXRMG variable to send an error message to the client. The rule can use the SDBINFO API function and pass or not pass the connection token as the second parameter.	Terminate the transaction task.

Criterion	Description	Default action
CPUTIME	A transaction task exceeded its maximum CPU time limit. This exception can be detected any time while the task is running. A rule for this event can take one or more of the following actions: • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread. This action prevents the exception from occurring again. Use the return value KILL to terminate the ODBC connection. No message is sent to the client. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	Terminate the transaction task.
IMSFAIL	An IMS task detected a failing IMS operation. This exception can occur for any type of IMS processing. The rule can use the SDBINFO function without passing the connection token as the second parameter.	Terminate the IMS operation, and reflect the error to the client task.
LOCKEXCLUSIVE	A transaction task exceeded its Db2 exclusive lock limit. A rule for this event can take one of the following actions: • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread. This action prevents the exception from occurring again. Use the return value KILL to terminate the ODBC connection. No message is sent to the client. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	Terminate the transaction task.

Criterion	Description	Default action
LOCKSHARE	A transaction task exceeded its Db2 share lock limit. A rule for this event can take one of the following actions: • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread. This action prevents the exception from occurring again. Use the return value KILL to terminate the ODBC connection. No message is sent to the client. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	Terminate the transaction task.
LOCKUPDATE	A transaction task exceeded its Db2 update lock limit. A rule for this event can take one of the following actions: • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread. This action prevents the exception from occurring again. Use the return value KILL to terminate the ODBC connection. No message is sent to the client. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	

Criterion	Description	Default action
LOGFAILURE	A Db2 database exceeded a pending logging requests limit. This exception can be detected at any time. A rule for this event can take one of the following actions: • Use the return value IGNORE to ignore the exception. This action preserves the contents of the pending request queue and prevents error messages from being issued. • Use the return value CLEAR to clear the pending request queue, release all associated storage, and send an error message that contains the number of cleared requests to the system console. Modify the limit so that the exception does not occur again.	

Criterion	Description	Default action
PERSQLCPU	A transaction task exceeded its per-SQL-statement CPU time limit. This exception is detected only by SQL operations that the server runs, for example for /*EXESQL rules. It is not detected when a user-written high-level language (HLL) program invokes long-running SQL operations. A rule for this event can take one of the following actions: • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread so that the exception does not occur again. • Use the return value KILL to terminate the ODBC connection. • Use the return value IGNORE to ignore the exception. • Modify the limit for the current thread. This action prevents the exception from occurring again. Use the return value KILL to terminate the ODBC connection. No message is sent to the client. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	Terminates the transaction.
PGMDURATION	An RPC stalled or was put it into an indefinitely long wait state. A rule for this event can take one of the following actions: • Examine the problematic program name and return no value, in which case the default action is taken. • Return the value IGNORE, which allows the problematic task and the RPC task to continue. Use the EXC.EXXDTMLM variable to modify the limit.	If no rule is enabled to handle the exception or if no return value is specified, the default action is to cancel the problematic task and clear the RCP program.

Criterion	Description	Default action
RPCENQUEUE	A transaction task detected that a client task exceeded its RPC enqueue time limit. This exception can be detected at any time. A rule for this event can take one of the following actions: • Return the value IGNORE to ignore the exception. • Modify the time limit for the current thread. • Return the value KILL to terminate the ODBC connection. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	
RTMONITOR	The application exceeded the client response time. This exception is detected only for ODBC connections.	None
SESSIONFAILURE	A transaction task detected that a client task exceeded the session failure limit. This exception can be detected at any time. A rule for this event can take one of the following actions: • Return the value IGNORE to ignore the exception. • Modify the time limit for the current thread. • Return the value KILL to terminate the ODBC connection. No message is sent to the client.	Terminate the ODBC client task.
SQLFAIL	A transaction task detected that a SQL statement failed. When a failure occurs, a negative SQL code is set. Only SQL operations that the server runs, such as for /*EXECSQL rules, detect this exception. The exception is not detected when a user-written high-level language (HLL) program invokes a long-running SQL operation.	Returns the SQL error code to the transaction task.

Criterion	Description	Default action
TIMERONLIMIT	A transaction task detected that a prepare returned a timer-on value that exceeds the limit. Only SQL operations that the server runs, such as for /*EXECSQL rules, detect this exception. The exception is not detected when a user-written high-level language (HLL) program invokes a prepare. A rule for this event can take one of the following actions: • Return the value ALLOW, which allows the exception. • Modify the limit. • Return the value REJECT, which terminates the SQL statement, and use the EXC.OPERMG variable to return an error message to the client. The rule can use the SDBINFO function without passing the connection token as the second parameter.	
WAITTIME	A transaction task exceeded the wait time limit. This exception can be detected at any time. A rule for this event can take one of the following actions: • Return the value IGNORE to ignore the exception. • Modify the limit. • Return the value KILL to terminate the ODBC connection. No message is sent to the client. The rule can use the SDBINFO API function and must pass the connection token as the second parameter. The connection token is required to identify the task that has the exception, rather than the current task.	
ZSQLALLIMSSEGMENTS	SQL Solution determined that a SQL statement causes all IMS segments that are specified as tables to be read because the child segments that are being joined are not constrained. The query does not specify the CHILD_ID and PARENT_ID columns in the WHERE clause.	Allow or terminate the SQL statement, which is based on the value of the SQLENGDFLTEXCACTION parameter.

Criterion	Description	Default action
ZSQLFULLDBREAD	SQL Solution determined that a SQL statement causes all database source records to be read because the subtable query is not constrained. The query does not specify the CHILD_KEY and PARENT_KEY columns in the WHERE clause.	Allow or terminate the SQL statement, which is based on the value of the SQLENGDFLTEXCACTION parameter.
ZSQLINCKEYBEGINNING	SQL Solution determined that only the beginning of an incomplete key was specified for one of the tables in a query. This situation might occur when multiple columns comprise the key and the query that is specified only the beginning columns. This situation is acceptable for VSAM access, but it might incur additional overhead for IMS access.	Allow or terminate the SQL statement, which is based on the value of the SQLENGDFLTEXCACTION parameter.
ZSQLINCKEYPARTIAL	SQL Solution determined that only part of an incomplete key was specified for one of the tables in the query and that the beginning portion of the key was not specified.	Allow or terminate the SQL statement, which is based on the value of the SQLENGDFLTEXCACTION parameter.
ZSQLNOKEYCOLUMNS	SQL Solution determined that no key columns were specified in the WHERE clause. This situation causes the entire database to be read.	Allow or terminate the SQL statement, which is based on the value of the SQLENGDFLTEXCACTION parameter.
ZSQLNOWHERECLAUSE	SQL Solution determined that no WHERE clause was provided for a table. This situation causes the entire database to be read.	Allow or terminate the SQL statement, which is based on the value of the SQLENGDFLTEXCACTION parameter.

Variables for all EXC events

You can use the variables in the following table in any EXC rule:

Variable	Contents	Data type
EXC.OPEXACSR	The action string for the current exception. This string cannot be directly changed; however, the return value from some rules can change the action string. The following are valid values: • ACCEPT: Accept the current condition • IGNORE: Ignore the current condition • KILL: Kill the current client connection • ALLOW: Allow the current exception • NOACTION: Take no action • REJECT: Reject the current exception • TERMINATE: Terminate the current client connection	Character, read-only
EXC.OPEXCNTK	The connection token that is used to obtain information about the thread where the exception occurred. You must use this field for all exceptions that the Check Limits task detects. The connection token is passed as the second parameter of the SDBINFO function. The connection token is only needed if the EXC.OPEXINFO flag is set to 0 (zero).	Character, read-only
EXC.OPEXERMG	The error message field. This field can be modified to send messages to the application.	Character, read-write
EXC.OPEXINFO	A variable that indicates whether the SDBINFO function can be used by the EXC rule. Valid values are: • 0 (zero): SDBINFO cannot be used • 1: SDBINFO can be used	Character, read-only
EXC.OPEXSRID	The search ID field contains the criterion that triggers the current rule. The valid values are listed in the previous table.	Character, read-only

Variable	Contents	Data type
EXC.OPEXWAOK	A variable that indicates whether the EXC rule is allowed to perform operations that cause the current subtask to be placed in a waiting state. An example of such a task is issuing an I/O request. Valid values are: • 0 (zero): WAITS are not allowed • 1: WAITS are allowed	Character, read-only
EXC.USER	The user area is passed among all rules that are triggered for the same event.	Character, read-write

Variables for CPULIMIT events

Variable	Contents	Data type
EXC.EXCLSPLM	The CPU time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop checking the CPU time.	Character, read-write
EXC.EXCLCPVL	The CPU time value shows how much CPU time the task has used.	Character, read-only

Variables for IMSFAIL events

Variable	Contents	Data type
EXC.EXIMIMCD	The IMS code. This value is obtained from IMS.	Character, read-only

Variables for LOCKEXCLUSIVE events

Variable	Contents	Data type
EXC.EXXCTMLM	The exclusive lock time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop checking the CPU time.	Character, read-only
EXC.EXSHTMVL	The share lock time value shows long the current task has been holding a share lock.	Character, read-only

Variables for LOCKUPDATE events

Variable	Contents	Data type
EXC.EXUPTMLM	The update lock time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop checking the CPU time.	Character, read-only
EXC.EXUPTMVL	The update lock time value shows long the current task has been holding an update lock.	Character, read-only

Variables for LOGFAILURE events

Variable	Contents	Data type
EXC.EXLGPNLM	The pending request limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop checking the limit of all pending requests. There are two request limits: the warning limit and the failure limit. If the rule is triggered for a warning limit, only the warning limit can be changed. If the rule is triggered for a failure limit, only a failure limit can be changed.	Character, read-only
EXC.EXLGPNVL	The pending requests value shows the number of pending logging requests.	Character, read-only
EXC.EXLGSSNA	The database name is the Db2 subsystem that has too many pending logging requests.	Character, read-only

Variables for PERSQLCPU events

Variable	Contents	Data type
EXC.EXPQCPLM	The per-SQL-statement CPU time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop all per-SQL-statement time checking.	Character, read-write
EXC.EXPQCPVL	The CPU time value shows the amount of CPU time that the current SQL statement used.	Character, read-only

Variables for PGMDURATION rules

Variable	Contents	Data type
EXC.EXXDTMLM	The program duration time limit, in seconds. If the PGMDURATION rule returns IGNORE, which allows the RPC program to continue, each time that the limit is checked later, an exception occurs.To avoid raising additional exceptions, change this variable to increase the program duration limit, or set the variable to 0 (zero) to prevent additional events from being recognized. If the rule puts a new limit into effect, the new limit applies only to the in-flight RPC program execution for which the current exception was raised. The new limit is not retained in memory.	Character, read-only
EXC.EXXDTMVL	The duration time value shows how long, in seconds, the RPC program has been running.	Character, read-only
EXC.EXXDPGNA	The 8-byte name of the RPC program load module that is being run. For SQL CALL statements, the full procedure name from the SQL statement is unavailable when this exception is recognized. If no RPC rule matches the SQL CALL procedure name, the value of this variable is the first 8 characters of the procedure name. If a matching RPC rule contains a PROGRAM section, the value of the variable is the 8-byte load module name from the PROGRAM section of the RPC rule. In this case, the 8 characters might not match the leading characters of the CALL statement procedure name.	Character, read-only

Variables for RPCENQUEUE rules

Variable	Contents	Data type
EXC.EXNQTMLM	The RPC enqueue time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop all RPC enqueue time checking.	Character, read-write

Variable	Contents	Data type
EXC.EXNQTMVL	The RPC enqueue time value, which shows how long the current task has been holding a PRC enqueue.	Character, read-only

Variables for RTMONITOR rules

Variable	Contents	Data type
EXC.EXCRTGRT	The client response time goal, which shows the acceptable response time.	Character, read-only
EXC.EXCRTMMI	The actual client response time for the transaction that produced the exception.	Character, read-only
EXC.EXCRTRTR	The total number of client response time records.	Character, read-only
EXC.EXCRSRTR	The sum of the total response time for all records.	Character, read-only
EXC.EXCRTMGR	The total number of client response time records that missed the response time goal.	Character, read-only
EXC.EXCRSMGR	The sum of the total response time for the records that missed the response time goal.	Character, read-only
EXC.EXCRIPAD	The IP address.	Character, read-only
EXC.EXCRUSID	The user ID.	Character, read-only
EXC.EXCRAPNM	The application name.	Character, read-only

Variables for SESSIONFAILURE rules

Variable	Contents	Data type
EXC.EXSETMLM	The session failure time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop all RPC enqueue time checking.	Character, read-write
EXC.EXSETMVL	The session failure time value, which shows how long the current task has been processing on behalf of a client.	Character, read-only

Variables for SQLFAIL rules

Variable	Contents	Data type
EXC.EXSQSQCA	The SQLCA is built by prepare and is provided as a single binary data area.	Character, read-only
EXC.EXSQSQCD	The SQL code that is obtained from the SQLCA.	Character, read-only
EXC.EXSQSQSR	The SQL statement that failed.	Character, read-only

Variables for TIMERONLIMIT rules

Variable	Contents	Data type
EXC.EXTMSQCA	The SQLCA is built by prepare and is provided as a single binary data area.	Character, read-only
EXC.EXTMSQSR	The SQL string that was prepared	Character, read-only
EXC.EXTMTMLM	The timer-on limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop all timer-on checking.	Character, read-only
EXC.EXTMTMVL	The timer-on value shows the timer-on value that is returned by prepare.	Character, read-only

Variables for WAITTIME rules

Variable	Contents	Data type
EXC.EXWATMLM	The wait time limit. This variable can be modified to prevent the exception from occurring again. Set the variable to 0 (zero) to stop all wait time checking.	Character, read-write
EXC.EXWATMVL	The wait time value, which shows how long the current task has been waiting for a request from a client.	Character, read-only

Global variable (GLV) events

If the SEFGLVEVENTS startup parameter is set to YES, a global variable event is generated when the value of a global variable is updated.

The default state of the SEFGLVENTS startup parameter is NO. To enable global variable events, set the parameter to YES. Then, if a REXX program updates a global variable, the SEF attempts to locate a rule that has the same name as the global variable. Enabling GLV events has an impact on the virtual storage that the subsystem uses and that rules are not triggered by high-level language (HLL) programs.

The following is the format for the header statement:

```
/*GLV criterion
```

where *criterion* is the name of the global variable.

A REXX process section is required. Return values are ignored. The global variable value is updated, regardless of the return value.

When a global variable event occurs, the system extracts information about the event and creates the following variables, which are instantiated when the rule is scheduled to run:

Variable	Contents	Data type
GLV.NAME	The 1- to 50-byte name of the global variable that triggered the rule.	Character, read-only
GLV.NEWVALUE	The value of the global variable, after it was changed. The standard REXX definitions apply to variables that were never previously referenced or were dropped.	Character, read-only
GLV.OLDVALUE	The value of the global variable, before it was changed. The standard REXX definitions apply to variables that were never previously referenced or were dropped.	Character, read-only
GLV.PROGRAM	The name of the program or rule that updated the global variable.	Character, read-only
GLV.TEXT	A text message that describes the event. The string, which is truncated at 100 bytes, includes the GLV.NAME, GLV.PROGRAM, GLV.OLDVALUE, and GLV.NEWNAME values.	Character, read-only
GLV.USER	An 8-byte field for communicating between rules that are triggered for a single event. Use this field to pass information between multiple rules. This field is initialized to binary zeroes.	Character, read-write

Remote procedure call (RPC) events

An RPC event occurs when then name of a remote procedure call matches the criterion of a rule.

For example, the following CALL statement could trigger an RPC rule:

```
CALL ALL ('LITERAL1', 'LITERAL2')
```

Use RPC rules for the following purposes:

Allow RPCs to be written in REXX
: REXX RPCs are not as versatile as those written in C, PL/1, COBOL, and Assembler. However, REXX RPCs can write SMF records and access host information and return it to the application.

Reject the current RPC statement
: To reject the current RPC statement, set the return value to REJECT. To send a message to the client, set the RPC.MESSAGE variable. If the current RPC statement is rejected, set the RPC.CODE variable to a negative value, which indicates failure.

Accept the current RPC statement
: To accept the current RPC statement, set the return value to ACCEPT. To send a message to the client, set the RPC. MESSAGE variable. If the current RPC statement is accepted, set the RPC.CODE variable to a positive value, which indicates success.

When an RPC event occurs, the system extracts information about the event and creates the following rule variables. These variables are instantiated when the rule is scheduled for execution.

Criterion	Variable	Contents	Data type
ALL	RPC.CODE	Returns an RPC code to the client.	Character, read-write
ALL	RPC.MESSAGE	Returns messages to the client.	Character, read-write
ALL	RPC.SEARCHID	The stored procedure name, which is extracted from the current SQL string. For example, for the following SQL statement, the search ID is `SAMPVSAM:CALL SAMPVSAM ()`	Character, read-only
ALL	RPC.TEXT	The SQL source string that invoked the current stored procedure. For example, for the following SQL statement, the text is `CALL SAMPVSAM ():CALL SAMPVSAM ()`	Character, read-only
ALL	RPC.USER	The user area is passed among all rules that are triggered for the same event.	Character, read-write

SQL events

A SQL event occurs when a SQL statement is processed.

A SQL rule runs before the SQL source is prepared. If a SQL source is modified, it is prepared or passed to run immediately after the SQL rule runs. Use SQL rules for the following purposes:

Modify a SQL source
: To modify a SQL source, add or modify a WHERE clause.

Reject a SQL statement
: To reject a SQL statement, use the REJECT return value. You can also use the SQL.MESSAGE to send a message to the client. If the SQL statement is rejected, set the SQL.CODE variable to a negative value. Otherwise, the value -1 is used as the SQL code.

Accept a SQL statement
: To accept a SQL statement, set the return value to ACCEPT. If the SQL statement is accepted, Db2 does not run it. Instead, the rule processes the statement. To send a warning or error message to the

client, use the SQL.MESSAGE variable. For warnings, a positive value. For failures, use a negative value. If the return code is ACCEPT and a non-zero value is set for the SQL.CODE variable, a message is sent to the client. If a message is not provided, a default message is constructed and sent.

When a SQL event occurs, the system extracts information about the event and creates the following variables. These variables are instantiated when the SQL rule is scheduled to run. You can write a SQL rule that accesses the following variables:

Criterion	**Variable**	**Contents**	**Data type**
ALL	SQL.CODE	The code to return to the client	Character, read-write
ALL	SQL.MESSAGE	The message to return to the client	Character, read-write
ALL	SQL.SEARCHID	The SQL verb that is extracted from the current SQL string	Character, read-only
ALL	SQL.TEXT	The actual SQL source	Character, read-only
ALL	SQL.USER	The user area that is passed among all rules	Character, read-write

Time-of-day (TOD) events

A time-of-day event occurs when the z/OS timer that is associated with a rule expires.

To specify the header statement, use the following syntax:

```
/*TOD todspec, interval, endspec, maxexecs
```

where:

- *todspec* is the date or time. You must specify either *todspec* or *interval*. Use one of the following formats to specify *todspec*:
  - *ddMMMyyyy*, where *dd* is a 2-digit integer (01 - 31) that represents the day of the month; *MMM* is a 3-character abbreviation for the month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC); and *yyyy* is a 4-digit year.
  - *yymmday*, where yy is a 2-digit year; mm is a 2-digit month; and *day* is the full name of a day of the week, for example, SUNDAY or MONDAY.
  - *hh*:*mm*:*ss*, where *hh*is a 2-digit integer (00 - 23) for the hour; *mm*is a 2-digit integer (00 - 59) for the minute; and *ss* is a 2-digit integer (00 - 59) for the seconds after the minute. The *ss* value is optional.
- *interval* is the amount of time to wait before running the rule again. You must specify either *todspec* or *interval*. Use the following format to specify the *interval*:
  - *n units*, where *n* is an integer that represents the number of times to run the rule, and *units* is the time to wait before running the rule again. For *units*, specify one of the following: DAY, DAYS, WEEK, WEEKS, HOUR, HOURS, MINUTE, MINUTES, SECOND, SECONDS.
- *endspec* is the time or date after which the rule stops running. This parameter is optional.
- *maxexecs* is an integer that represents the maximum number of times to run the rule. This parameter is optional.

Note: If you omit any parameter, code a comma in its place.

The value that is returned from a TOD rule has no special meaning.

When a TOD event occurs, the system extracts information about the event and creates the following variables. These variables are instantiated when the rule is scheduled for execution.

Criterion	Variable	Contents	Data type
ALL	TOD.NEXTFIRE	A value that indicates the next time that the rule runs. The following are valid values: • The date and time in *yyyy*/*mm*/*dd hh*:*mm*:*ss* format. • NONE if the rule will not run again.	Character, read-only
ALL	TOD.USER	An 8-byte field for passing information among multiple rules. This field is initialized to binary zeroes.	Character, read-write

Virtual table (VTB) events

Virtual table events are generated by the SQL Engine when a table name is found in an SQL statement. These events are only generated if the **SEFVTBEVENTS** startup parameter is set to allow them. The rules allow for creating virtual tables dynamically from a Data Mapping facility (DMF) model map and for modifying certain table values.

No keywords are defined for VTB event procedures. Only the SQL engine schedules execution of enabled VTB event procedures for each table name in an SQL statement. VTB event procedures allow you to modify information in the DMF map. VTB event procedures make it possible to access multiple data sets using one DMF map by creating alias maps using a model map. Each alias map can specify a different data set name. The model map must be a map that is created by using DMF.

Only the event procedure criterion value is allowed (and *must* be present).

To specify the header statement, use the following syntax:

```
/*VTB criterion
```

where:

- *criterion* is the criterion value for VTB event procedures. This *criterion* is one of the two event types that are shown in the following table.

Each VTB event procedure has access to server-wide global variables.

In addition, VTB-specific variables are created before the VTB event procedure is invoked. The variables that are created differ depending on the criterion.

Criterion	Variable	Contents	Data type
Any criterion	VTB.USER	The user area is passed between all event procedures that fire for the same event.	Read-write

Criterion	**Variable**	**Contents**	**Data type**
Any criterion	VTB.OPTBSRID	The search id field contains the criterion used to fire the current event procedure. The format of the criterion is the string ‘MODIFYTABLE.’ followed by the table name found in the SQL statement.	Character Read-only
Any criterion	VTB.OPTBTBNA	The 1 to 128-character table name from the SQL statement.	Character Read-only
MODIFYTABLE.*tablename*	VTB.OPTBMTNA	Set the model table name. This is the 1 to 50-character name of a DMF map that will be used to create a virtual table with the alias name *tablename*	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBMRDI	Disable MapReduce. Set this value to 1 to disable map reduce. Setting this value to 0 has no effect. VTB.OPTBMRDI and VTB.OPTBMREN are mutually exclusive.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBMREN	Enable MapReduce. Set this value to 1 to enable map reduce. Setting this value to 0 has no effect. VTB.OPTBMREN and VTB.OPTBMRDI are mutually exclusive. Enabling MapReduce requires that the MapReduce feature is enabled.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBMRTC	Set the number of MapReduce threads to use.	Character, write

Criterion	Variable	Contents	Data type
MODIFYTABLE.*tablename*	VTB.OPTBFLAT	Flatten this table. Set this value to 1 to flatten the table. All columns and occurrences are returned in a single table Setting this value to 0 has no effect. VTB.OPTBFLAT and VTB.OPTBSUBT are mutually exclusive.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBSUBT	Create subtables. Set this value to 1 to create subtables Columns that are part of an occurs or occurs-depending-on are returned as separate tables. Setting this value to 0 has no effect. VTB.OPTBFLAT and VTB.OPTBSUBT are mutually exclusive.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBCLSQ	Clear sequential data set map related fields. Set this value to 1 to clear the data set member name, pre-write exit name, and post read exit name. Setting this value to 0 has no effect. The fields are cleared before any other variables are processed.	Character, write

Criterion	Variable	Contents	Data type
MODIFYTABLE.*tablename*	VTB.OPTBCLCI	Clear VSAMCICS map related fields. Set this value to 1 to clear the pre-write exit name, post read exit name, CICS file control table entry names, CICS connection name, and CICS transaction name fields. Setting this value to 0 has no effect. The fields are cleared before any other variables are processed. Clearing those fields cause a VSAMCICS file to be processed as a native VSAM file.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBCLAD	Clear Adabas map related fields. Set this value to 1 to clear the database ID, file number, and subsystem name fields. Setting this value to 0 has no effect. The fields are cleared before any other variables are processed.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBCLD2	Clear Db2 map related fields. Set this value to 1 to clear the table name, subsystem map name, table creator name, plan name, and user ID fields. Setting this value to 0 has no effect. The fields are cleared before any other variables are processed.	Character, write

Criterion	Variable	Contents	Data type
MODIFYTABLE.*tablename*	VTB.OPTBCLIM	Clear IMS DB map related fields. Set this value to 1 to clear the segment name, DBD name, and PSB name fields. Setting this value to 0 has no effect. The fields are cleared before any other variables are processed.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBCLIV	Clear IMS view map related fields. Set this value to 1 to clear the segment name, DBD name, and PSB name fields. Setting this value to 0 has no effect. The fields are cleared before any other variables are processed.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBDSNA	Set the 1 to 44-character VSAM or sequential data set name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBMEMA	Set the 1 to 8-character sequential data set member name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBPRWR	Set the 1 to 8-character VSAM, VSAMCICS, or sequential data set pre-write exit name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBPSRD	Set the 1 to 8-character VSAM, VSAMCICS, or sequential data set post read exit name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBVSBF	Set the 1 to 8-character CICS file control table entry name for the base file.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBCONN	Set the 1 to 4-character CICS connection name.	Character, write

Criterion	Variable	Contents	Data type
MODIFYTABLE.*tablename*	VTB.OPTBCITR	Set the 1 to 4-character CICS transaction name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBADBI	Set the Adabas database ID (DBID) number.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBAFNR	Set the Adabas file number.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBSUBS	Set the 1 to 4-character Adabas subsystem name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBD2TN	Set the 1 to 128-character Db2 table name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBD2SN	Set the 1 to 50-character Db2 subsystem map name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBD2TC	Set the 1 to 8-character Db2 table creator ID.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBD2PN	Set the 1 to 8-character Db2 plan name.	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBIMSN	Set the 1 to 8-character IMS DB segment name	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBIMDN	Set the 1 to 8-character IMS DB DBD name	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBPSB	Set the 1 to 8-character IMS DB PSB name	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBIVSG	Set the 1 to 8-character IMS view segment name	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBIVDB	Set the 1 to 8-character IMS view DBD name	Character, write
MODIFYTABLE.*tablename*	VTB.OPTBIVPS	Set the 1 to 8-character IMS view PSB name	Character, write

Criterion	Variable	Contents	Data type
GETALIASES.*tablename*	VTB.OPTBLIST	Set a list of 1 to 50-character table names that are the aliases of map *tablename*. There is room for up to 637, 50-character alias names that are separated by a blank. More alias names are possible if they are shorter.	Character, write

Host commands

Use host commands to retrieve output information from a specified host environment.

DISPLAY command

Use the DISPLAY command to display information about all connected users.

Displaying basic information

Use the following syntax to display basic information about all connected users:

```
"DISPLAY REMOTE USERS(*)"
```

This command displays the following information about each connected user:

- ACTUAL BLOCK ADDRESS
- APPLICATION NAME
- CONNECTION ID
- DB2 SUBSYSTEM
- HOST NAME
- ICUV PATH ID
- IP ADDRESS
- LINK TYPE
- LOCAL IP PORT NUMBER
- REMOTE IP PORT NUMBER
- SOCKET NUMBER
- TRUSTED HOST
- USER ID
- TASK TCB ADDRESS
- TRUSTED HOST
- USER ID

Displaying additional information

Use the following syntax to display additional information about all connected users:

```
"DISPLAY REMOTE USERS(*) VERBOSE"
```

This command provides the following additional information about each connected user:

- ACEE SOURCE
- BUFFER FUNCTION CODE
- COMPRESSED SEND AMOUNT
- COMPRESSED TOTAL BYTES RECEIVED
- CPU TIME
- CUMULATIVE COMPRESSION
- CUMULATIVE RECEIVED COMPRESSION
- CURRENT COMPRESSED RECEIVED AMOUNT
- CURRENT RAW RECEIVED AMOUNT
- CURRENT STATE
- DB2 PLAN NAME
- DB2 REQUESTING SITE
- DB2 THREAD TOKEN
- DOMAIN NAME
- ELAPSED TASK TIME
- EXTENDED USER ID
- GENERIC USER ID
- HOST TIME
- INTERNAL NAME
- LAN USER ID
- LOCKS HELD
- MODULE NAME
- ODBC DRIVER DATE
- ODBC DRIVER VERSON
- PROGRAM NAME
- RAW BYTES RECEIVED
- RAW BYTES SENT
- RAW RECEIVED COMPRESSION
- RAW SEND AMOUNT
- RAW SEND COMPRESSION FACTOR
- SQL CODE
- SQL COUNT
- SQL CURSOR NUMBER
- SQL REASON CODE
- SQL RETURN CODE
- SQL STATEMENT NUMBER
- SQL STATEMENT TYPE
- STATE DURATION
- TELEPROCESSING TIME
- TELEPROCESSING TIME PERCENTAGE
- TOTAL RAW BYTES SENT
- USER PARAMETER
- WLM ENCLAVE COUNT

- WLM ENCLAVE CPU TIME

API functions for rules

AVZVALUE API function

Use the AVZVALUE function to manipulate global variables.

For example, use the AVZVALUE function to use compound symbols as a type of database. Use this function in a rule that performs special interrogation or serialization processing.

Under normal circumstances, you can use a REXX language statement to reference or set the value of a global variable. The following code shows an example of using a REXX statement to

```
SAVENAME = GLOBAL.COMPANY.NAME
GLOBAL.COMPANY.NAME = "Keroct Software"
GLVEVENT.MYDATA = "ABC"
```

Syntax

```
val = AVZVALUE(derivedname, actioncode, newval, oldvar)
```

where:

- *derivedname* is the name of the symbol that receives the action. When you use this parameter without quotation marks, simple symbols (case sensitive) following the stem are replaced by their values.
- *actioncode* is the action to take on the symbol.
- *newval* is the new value to assign to the symbol.
- *oldval* is the value of the symbol before the action is taken.

Return values

AVZVALUE returns a value from the function call, and for some action codes, places information in the external data queue.

Action codes

The following table describes the actions that are performed for each action code and the values that are returned.

Table 37. Action Codes and return values

Action code	Descriptin	Return value	Description
A (Add)	Adds a number, which is specified by increment, to the existing compound symbol given by *derivedname*. All references to the compound symbol are serialized during the add operation, so you can use this function to increment a counter that is set by concurrent tasks.	`val = AVZVALUE (derivedname'A' increment)`	Returns 1 (true) if the comparison finds the pre-action value to be equal to the old value and the compound symbol was updated. Returns 0 (false) if the comparison finds unequal values and does not update the value of the compound symbol. Does not change the external data queue.

Table 37. Action Codes and return values (continued)

Action code	**Descriptin**	**Return value**	**Description**
C (Compare and update)	Verifies the value of a compound symbol and then updates its value. Safely updates global symbols that more than one rule uses or global symbols that multiple copies of the same rule might access and update. Serializes the compare and update operations for global values.	`val = AVZVALUE (derivedname,'C', newval,oldval)`	Returns 1 (true) if the comparison finds the pre-action value to be equal to the old value and the compound symbol was updated. Returns 0 (false) if the comparison finds unequal values and does not update the value of the compound symbol. Does not change the external data queue.
D (Drop)	Drops the compound symbol that is specified by *derivedname*. Resets the compound symbol to its uninitialized value or derived name. If *derivedname* specifies a stem, all compound symbols that belong to that stem are dropped and the virtual storage that is allocated to them is released. All other references see the compound symbol as it existed before the drop operation started or as it is after the drop operations finishes.	`val = AVZVALUE (derivedname,'D')`	Returns the value of *derivedname*. Does not change the external queue.
E (Existence)	Determines whether a global variable exists.	`val = AVZVALUE (derivedname,'E')`	Returns one of the following values for the status of the global variable: • I: Initialized • U: Uninitialized. The variable exists in storage, but it is uninitialized so it is set to the value of its name. • N: Does not exist. The variable does not exist in storage. Does not change the external data queue.

Table 37. Action Codes and return values (continued)

Action code	**Descriptin**	**Return value**	**Description**
F (Find)	Determines whether a global variable exists. The maximum length for a string pulled from the external data queue is 350 bytes. Longer strings are truncated.	`val = AVZVALUE (derivedname,'F')`	Returns one of the following values for the status of the global variable: • I: Initialized • U: Uninitialized. The variable exists in storage, but it is uninitialized so it is set to the value of its name. • N: Does not exist. The variable does not exist in storage. When the return value is I or U, the value of the node is returned in the external data queue.

Table 37. Action Codes and return values (continued)

Action code	**Descriptin**	**Return value**	**Description**
I (Information)	Returns information about all of the immediate subnodes of the *derivedname*.	`val = AVZVALUE (derivedname, 'I')`	For each subnode, places two lines in the external data queue. The first line contains the next segment of the *derivedname*. The second line contains the following information about the *derivedname*: • Word 1, length 8: Number of subnodes under this node. • Word 2, length 8: Create date, in the form yy/mm/dd. • Word 3, length 8: Create time, in the form hh:mm:ss. • Word 4, length 17: Create rule or program name. • Word 5, length 8: Create job name, task name, or TSO ID. • Word 6, length 8: Last modification date. • Word 7, length 8: Last modification time. • Word 8, length 17: Last modification rule or program name. Does not return partially updated symbol names.
L (List)	Lists the derived name of each subnode of the *derivedname*.	`val= AVZVALUE (derivedname, 'L')`	Returns the number of subnodes that are listed in the external data queue. Returns dropped symbols, but does not return removed symbols.

Table 37. Action Codes and return values (continued)

Action code	Descriptin	Return value	Description
O (Obtain)	Obtains the value of a global variable.	`val = AVZVALUE (derivedname, 'O')`	Returns the value of a global variable. If the global variable does not exist, returns an error. Does not change the external data queue.
R (Remove)	Removes the specified node and all of its subnodes. After a node is removed, it ceases to exist.	`val = AVZVALUE (derivedname, 'R')`	Returns the number of subnodes that were removed. Does not change the external data queue. Does not allow other accessories of compound symbols to see partially updated symbols.
S (Subtree)	Lists the entire global variable name of all subnodes of the *derivedname*.	`val = AVZVALUE (derivedname,'S')`	Returns the entire global variable name of all of the subnodes in the external data queue. Returns the number of subnodes that exist, as listed in the external data queue. Does not return partially updated symbol names.
T (Subtree and information)	Lists the entire global variable name and all subnodes of the *derivedname*.	`val = AVZVALUE (derivedname,'S')`	Returns the entire global variable name and two lines for each subnode in the external data queue. The first line contains the next segment of the *derivedname*. The second line contains the information., as described for the Information code, for each *derivedname*. Does not return partially updated symbol names.

Table 37. Action Codes and return values (continued)

Action code	Descriptin	Return value	Description
U (Update)	Assigns *newval* as the value of the compound symbol that is specified by *derivedname*. If the compound does not exist, the compound is created and assigned the new value. Use Update to prevent others who access compound symbols from seeing partially updated symbols.	`val = AVZVALUE (derivedname,'U',newval)`	Returns the variable that is specified by *newval*. Does not change the external data queue.
V (Value)	Returns the value of the specified compound symbol. Use Value to prevent the issuer of SDVALUE from seeing partially updated symbols.	`val = AVZVALUE (derivedname,'V')`	Returns the current value of the node. If the node does not exist, it is created but it is not assigned a value. Instead, it is given the same value as its name. Does not change the external data queue.

AVZINFO API function

The AVZINFO function retrieves information about the Data Virtualization Manager server subsystem.

The syntax for the AVZINFO function is the following:

```
var=AVZINFO(arg1[,arg2])
```

where *arg1* is a parameter from the following table, and *arg2* is the connection token, which is optional.

The function always returns a return value. If the value requested is not valid for the environment, a NULL string is returned.

Parameter	Return value
ASID	Returns the address space identifier (ASID) as a 2-byte binary value when invoked using the program API. Returns the ASIDD as a 4-byte value when invoked from REXX.
BYTES	Returns the number of saved bytes.
CLOCK	Returns the current time-of-day (TOD) clock value as an 8-byte binary value. This is the unadjusted STCK value.
CONNECTID	Returns the unique connection ID value.
CPUDELTA	Returns the 8-byte task CPU time delta value.
CPUTIME	Returns the 8-byte task CPU time value.
DB2PLAN	Returns the name of the Db2 plan.

Parameter	Return value
DB2SUBSYS	Returns the name of the Db2 subsystem.
EVENTTYPE	Returns the type of event that is associated with the rule or program.
HOSTDOMAIN	Returns the host (server) domain that is associated with the current request.
HOSTNAME	Returns the host name (client) associated with the current request.
IPADDRESS	Returns the fully formatted IP address for the current request in the form 10.17.16.164.
JOBNAME	Returns the z/OS job name that is related to the current primary address space.
LASTCONNECTID	Returns the last connection ID used on the current link.
LASTUSERID	Returns the last user ID used on the current link.
LINKTYPE	Returns the link type for the current request.
LU	Returns the LU name for the current request.
MAINPGM	Returns the name of the main REXX program or rule.
MODE	Returns the mode name for the current request.
ODBCDATE	Returns the compile date of the .NET Client (ODBC).
ODBCVERSION	Returns the version of the .NET Client (ODBC).
PRODUCT	Returns the product identification string.
PRODUCTSTATUS	Returns the current product status.
PROGRAM	Returns the name of the REXX program or rule.
ROWS	Returns the number of source rows.
SEFFEATURE	Returns a single blank if the Server Event Facility (SEF) is not enabled.
SUBSYS	Returns the accessed subsystem ID from the current OPMS image.
SUBSYSASID	Returns the ASID of the active subsystem from the real OPMS as a 2-byte binary value when invoked by using the program API and as a 4-byte value when invoked from REXX.
SMFID	Returns the SMF ID.
TASKTYPE	Returns the task type.
TRANSTYPE	Returns the transaction program type.
USERID	Returns the user ID value.
USERPARM	Returns the user parameter string from the client.

Parameter	Return value
VERSION	Returns, as a string, the version of the product subsystem under which the rule or program is running.

Examples

The following call sets the REXX variable, IPA, to the fully formatted TCP/IP address of the client program:

```
IPA = AVZINFO(IPADDRESS)
```

The following call sets the variable *USER* to the user ID value of the connection that caused the exception. In this example, EXC.OPEXCNTK, which contains the connection token, is used to obtain the user ID because the exception rule runs under the OPCKLM (check limits) task, not the user connection task:

```
USER = AVZINFO(USERID,EXC.OPEXCNTK)
```

AVZECURE API function

The AVZECURE function performs security-authorization processing.

Verify data set access

To verify that the current user has authorization to access a data set, use the following syntax:

```
var = AVZECURE('D','dsname','accesstype','volser')
```

where:

- *dsname* is the name of the data set.
- *accesstype* is the type of data set access to verify. If you do not specify a type, READ access is the default. Valid values are:
  - A: Verify ALTER access.
  - C: Verify CONTROL access.
  - R: Verify READ access.
  - U: Verify UPDATE access.
- *volser* is the volume serial number to validate. If you do not specify a volser, the parameter is blank, by default.

The function returns a message that indicates whether access is allowed.

Retrieve logon ID field data

To retrieve security subsystem information from the current user's ACEE, use the following syntax:

```
var = AVZECURE('F','fieldname')
```

where *fieldname* is one of the fields in the following table:

Field	Description	Field format
ALTER	Alter authority flag	Bit
APPLICATION	Application name	Character
APPLICATIONDATA	Application data	Character
APPLICATIONLEVEL	Application level	Binary
AUDITOR	Auditor attribute	Bit

Field	Description	Field format
AUTOMATIC	Automatic attribute	Bit
CLASSAUTHORIZATIONS	Class authorizations	Binary
CONTROL	Control authority flag	Bit
DATE	Date	RACINT date
DEFINEUSERS	Authorized to define users	Bit
GROUP	Contents of the ACEE group field	Character
GROUPLIST	A list of groups	Character
GROUPLISTCONTAINS	Group list contents flag	Bit
INSTALLATIONDATA	Contents of the installation data field	Character
LOG	Logging on for most operations	Bit
NONE	None authority flag	Bit
OPERATIONS	Operations attribute	Bit
PORTOFENTRYDATA	Port of entry data	Character
PORTOFENTRYLEVEL	Port of entry level	Binary
PRIVILEGED	Started task with privileged flag	Bit
PROTECTDASD	Authorized to protect DASD	Bit
PROTECTTAPE	Authorized to protect tape	Bit
PROTECDTTERMINALS	Authorized to protect terminals	Bit
RACF®	RACF-defined user flag	Bit
READ	Read authority flag	Bit
SPECIAL	Special attribute	Bit
STCNAME	Started task name	Character
SURROGATEUSERID	Surrogate user ID	Character
TERMINAL	Terminal ID	Character
UPDATE	Update authority flag	Bit
USERDATA	Contents of the user data field	Character
USERID	Contents of the ACEE user ID field	Character
USERNAME	User name field	Character
VERSION	ACEE version code	Binary

The following conversions occur, based on the field format:

- Binary fields are converted to signed decimal values without leading zeroes or blanks. The number zero is returned as 0.
- Character fields are returned as is. If a character field name exceeds the maximum allowed string length, it is truncated to the Data Virtualization Manager configuration/REXX-defined maximum string length.

- Date fields are converted to the format *yyyy/mm/dd*. Leading zeros are retained so that the result is always 10 non-blank characters. A date field that contains zero is returned as ****/**/**.
- Bit fields are converted to 0 (false or off) or 1 (true or on).
- The GROUPLIST field inquiry returns an integer that represents the number of entries in the group list. Each group name is returned as a separate entry in the external data queue.

Request security product information

To retrieve information about the security product, use the following syntax:

```
var = AVZECURE('i','name')
```

where *name* is one of the values in the following table:

Value	Return value
MODE (Valid only for systems that run ACF2)	Returns one of the following ACF2 operating modes: ABORT, LOG, OFF, WARN, QUIET.
PRODUCT	Returns the name of the security product or the message UNKNOWN SECURITY PRODUCT.
RELEASE	Returns the release and version number for the security product.

If the information cannot be obtained, a NULL string is returned.

Verify access to a generalized resource

To verify that the current user has access to a generalized resource, use the following syntax:

```
var = AVZECURE('R', class, resource, requestcode)
```

where:

- *class* is the generalized resource class name or for ACF2, the type name.

  Note: Rules that verify access to resources use SAF processing. If you use ACF2, you must define the ACF2 resource type as a SAF class name.
- *resource* is the 1- to 39-byte resource entity name.
- *requestcode* is the type of access to verify. If you do not specify a request code, READ access is the default. The following are valid values:
  - A: Verify ALTER access..
  - C: Verify CONTROL access.
  - R: Verify READ access.
  - U: Verify UPDATE access.

If access to the resource is allowed, the string ALLOW is returned. Otherwise, an error message is returned.

Verify a user ID and password

Use the following syntax to verify the user ID and password. If the password is valid, the user is logged on to the system. This API call is valid only for ATH events.

```
var = AVZECURE('P','userid','password','newpassword')
```

where:

- *userid* is the user ID to validate.

- *password* is the password that is associated with the user ID.
- *newpassword* is the new password to associate with the user ID.

If you omit the *newpassword* parameter, the user ID and password are validated. If you specify the *newpassword* parameter, the password is changed.

If the password is correct, the return value is the string ALLOW. If the password is incorrect, an error message is returned. For ACF2, the counter for invalid password violation for the specified user ID is incremented for each failed attempt.

Use an implied password to validate a user ID

This request causes the specified user ID to be validated. If the password is valid, the user is logged on to the system. The password is not specified on the function call. Instead, the initial inbound transaction request transmits the password. Use this function to perform custom security checks without making the clear text password available to the procedure. This API call is valid only for ATH events.

Use the following syntax to use an implied password to validate a user ID:

```
var = AVZECURE('PI', 'userid', 'newpassword')
```

where:

- *userid* is the user ID to validate.
- *newpassword* is the new password to associate with the user ID.

If you omit the *newpassword* parameter, the function uses the implied password to validate the user ID. If you specify *newpassword*, the function changes the password. If the password is correct, the return value is the string ALLOW. If the password is incorrect, an error message is returned. For ACF2, the counter for invalid password violation for the specified user ID is incremented for each failed attempt.

AVZSUBMIT API function

Use the AVZSUBMIT function to submit JCL to the internal reader and return the JES2 or JES3 job ID for each submitted job.

The AVZSUBMIT function can be invoked as a function reference, which returns its result to the point of invocation, or as a REXX CALL statement. There is no corresponding TSO/E REXX or high-level language (HLL) API interface.

- The JCL statements read from the input stream can be any size; however, each individual statement is extended or truncated to be 80 bytes when submitted through the internal reader.
- In cases where the JCL input stream is ASCII or UTF-8 encoded, for example, for POSTED input, the function converts the JCL stream to IBM-1047 EBCDIC. Only rudimentary UTF-8 support is available, so avoid including double-byte characters and ASCII characters above code point 0x7F.
- The function provides no editing and imposes no restrictions on the content and format of JOB statement names in the JCL that is submitted.
- To detect job boundaries, the function scans each JCL statement. The following situations indicate a job boundary:
  - The JCL statement begins with "//", followed by an uppercase EBCDIC Latin letter or one of the IBM 1047 EBCDIC characters "@", "$", or "#".
  - The prefix is followed by 0 - 7 Latin letters or numbers or the IBM 1047 EBCDIC characters "@", "$", or "#".
  - The next blank-delimited word is JOB. After this word is found, the scan stops parsing the statement.
  - The scan does not take into account quoted string boundaries that enclose continued PARM= operands and does not detect, honor, and process JCL statement continuations.
- Jobs that are submitted while a client user ID logon are in effect are given a USER attribute that matches the logon ID of the client subtask. If the JCL USER= operand of the JOB statement is present

and differs from the client task logon ID and PASSWORD= is not present, RACF surrogate user attribute assignment and authorization restrictions might be imposed.

- The AVZSUBMT function can be used only in REXX language rules. The function cannot be used in a rule that runs in cross-memory mode or one for which waiting for system services is inhibited. Areas where AVZSUBMT cannot be used or can be used only conditionally include the following:
  - AVZSUBMT cannot be used during enabling or disabling a rule, which occurs when the PHASE variable is not set to PROC.
  - AVZSUBMT cannot be used in CMD, GLV, and TYP rules.
  - To determine when AVZSUBMT can be used, an ATH rule can check the value of the ATH.OPAU13WA variable, and an EXC rule can check the value of the EXC.OPEXWAOK variable. If AVZSUBMT can be used, the variable is preset to 1.

Use the following syntax:

```
AVZSUBMIT(arg1, arg2, arg3, arg4 )
```

or

```
CALL AVZSUBMIT(arg1, arg2, arg3, arg4 )
```

where:

- *arg1* and *arg2* specify the location of the input JCL stream.
- *arg3* specifies the 1-character JES class to which the internal reader is allocated.

  reader is allocated.
- *arg4* is a string that specifies the type of tracing.

The following table lists the valid values for *arg1* and *arg2*:

Value	**arg1: Location of the JCL input stream**	**arg2**
STEM	The JCL is in a REXX stem variable array. The 0th entry in the array contains the count of entries. Entries 1 - *n* contain individual JCL statements.	The REXX variable stem name. The name must end with a period. Length 1- 12 character.
DSN	The JCL is in a z/OS data set.	A fully qualified z/OS data set name. The name can include a PDS(E) member name. Length 1-54 bytes.
DDN	The JCL is in a z/OS data set that is preallocated to a DD name.	The DD name. Length 1- 8 bytes.
PATH	The JCL is in a USS HFS file.	The fully qualified HFS path name of the file. Length 1 - 256 bytes.
POSTED	The JCL is received as a posted file entity over HTTP.	The index number, 1 to *n*, of the posted file entity in the received HTTP request. If this argument is omitted, the default value is 1.

arg3 is the 1-character JES class to which the internal reader is allocated. The character A - Z, 0 - 9, and * (asterisk) are valid. Use * to request the default job class. If you do not specify this parameter, * is the default.

arg4 is a string that is 1 - 5 bytes. Each character of the string must be Y or N to specify whether the corresponding trace function for that byte is enabled. The following table describes the byte positions and trace functions:

Byte position	Default	Trace function
1	Y	Trace JOB IDs that JES returns.
2	Y	Trace input source JCL.
3	Y	Trace the dynamic-allocation activity of the internal reader.
4	N	Trace writes to the internal reader.
5	N	Trace the decoding of posted data (conversion to EBCDIC).

Unless a REXX ERROR or FAILURE signal is generated because of a fault condition, *arg4* returns one of the following numeric results:

- 0: Successful completion
- 4: Parameterization error
- 8: Environmental error
- 12: System service error
- 16: ABEND condition that is trapped
- +100: If one or more jobs are submitted before a failure, the value +100 is added to a result. To determine the failure code, subtract 100.

JOBID. stem variables

The function uses a REXX DROP on all JOBID. stem variables during entry-processing and presets variables to the values shown in the following table. This reset operation occurs after initial parameter validation but before JCL processing. If the reset fails, the REXX `invalid symbol` signal is generated. After setup, unless a REXX signal is thrown, the JOBID.RC, JOBID.REASON, JOBID.0, and JOBID.*n* variables are set as described. All other JOBID. stem variables are undefined.

Variable	Description
JOBID.RC	Contains the same value as the evaluated RESULT of the function call or if a problem is detected before all other JOBID. stem variables are correctly set, contains a NULL string.JOBID.RC is set to a NULL string at entry, and setting this variable to the RESULT is the last action that the function takes before exit.
JOBID.REASON	When the function call ends with a non-zero RESULT, contains error text. This variable is set to a NULL string when the RESULT is zero.

Variable	Description
JOBID.0	Contains an integer that indicates the number of jobs that were found in the input JCL stream and successfully submitted to the internal reader. If no jobs were successfully submitted or if a system failure prevented the return of any job IDs during processing, this variable contains 0 (zero). If one or more jobs are submitted before a failure, this variable contains the number of submitted jobs for which IDs were returned.
JOBID.*n*	Contains the job ID that is assigned to the first through *n*th job in the submitted JCL stream. Valid job IDs are in the format JOB*xxxxx* or J*xxxxx*, where *xxxxx* is a system-assigned sequence number. Only the variables JOBID.1 through JOBID.*n*, where *n* is the numeric value that is assigned to JOBID.0 are set.

Chapter 7. Logging and tracing server information

System information that is unique to the Data Virtualization Manager server can be recorded in Server Trace log file(s), System Management Facility (SMF) data set records, and in Data Virtualization log (DB2) tables.

Use Server Trace to trace and log server events, and to help identify and troubleshoot Data Virtualization Manager server issues.

Use the System Management Facility (SMF) to record system resource usage information in SMF data sets.

Use DB2 logging to write out the total z/OS resource usage information into an intervals table for a specified time interval.

The information that you collect can be used for:

- Billing users
- Reporting reliability
- Analyzing the configuration
- Scheduling jobs
- Summarizing direct access volume activity
- Evaluating data set activity
- Profiling system resource use
- Maintaining system security

You can choose to use any combination of logging features. For example, SMF logging can be used together with Server Trace logging, or separately. When used together, only the SMF Record Subtypes 01, 02, 06, 09, 10, 18 and 19 records are available for logging into DB2 tables. In most cases, the fields that are recorded are identical. However, some fields may not be available in both SMF and Server Trace logs.

Server Trace

Use Server Trace to trace and log server events, and to help identify and troubleshoot Data Virtualization Manager server issues.

By default, for each event that occurs on the Data Virtualization Manager server, an entry is created and stored in the trace log.

You can view the trace log from the studio or from the Data Virtualization Manager server ISPF panels.

- To view the trace log from the Studio, open the Server Trace view from the studio **Window** menu, select **Show View**, and then select **Server Trace**.
- To view the trace log from the ISPF application, select option **B Server Trace** > **Server Trace Facility** > **SIS SSID:AVZS**.

  Note: To view the Server Trace information for a different server, replace **AVZS** with the appropriate subsystem name.

As a session runs, entries are created for the following types of events:

- SQL operations
- IMS calls
- CICS calls
- Communication events
- Thread attach and detach events
- RPC events

- Message events
- Errors and abends

When the Data Virtualization Manager server runs a SQL statement, multiple entries are created in the trace log and in the client log. Each log records the series of events from a different perspective.

For example:

A client that runs a SQL statement could record the following entries:

- `SEND event`
- `RECEIVE event`
- `SQL event` (The results are returned.)

While the Server Trace log records the following entries:

- `RECEIVE event` (Matches the client SEND event.)
- `SQL event` (The SQL statement that is sent to DB2®.)
- `SEND event` (Matches the client RECEIVE event.)

If you were to view a combined log of the SQL statement execution, it would appear in the order each event occurred. For example:

- `SEND event` (Client side.)
- `RECEIVE event` (Server side.)
- `SQL event` (Server side.)
- `SEND event` (Server side.)
- `RECEIVE event` (Client side.)
- `SQL event` (Client side.)

The Trace Browse consists of a large block of virtual storage that is used to back up active trace browse data. This block of virtual storage is subdivided into a status area, a configurable number of event blocks, and a series of vector tables. The entries are initially added to a buffer that is maintained in virtual storage. In general, the buffer can accommodate the complete history of all client and server processing for several days. The entries are written to disk (a VSAM data set) every "n" seconds, as set by the parameter `BROWSEINTERVAL` (`SERVER TRACE CHECKPOINT INTERVAL`).

The Data Virtualization Manager server supports multiple trace data set Archives. Using hierarchical storage management, you can maintain an unlimited history of entries.

You can configure the Instrumentation Server to route entries from multiple instances of the Data Virtualization Manager server in a sysplex, to a single repository; giving you with a global view of all activity in a single ISPF panel.

The SQL Trace program provides information about the SQL statements that applications issue. When you select the active session, SQL Trace uses the connection ID as criteria to obtain and display SQL entries from the trace log.

When the `TRACEEXTERNTRACEDATA` parameter is set to YES, the Trace Data from the driver connection is sent to the server in the Servers Trace Browse area.

Displaying and navigating log entries

Use the Server Trace panel to view, navigate, and manage the log entries that display.

About this task

By default, the Server Trace panel displays all log entries. To view a subset of the log entries, you can filter on the results, use labels, and create a profile. If the Data Virtualization Manager configuration is running on a zIIP server, entries that are related to work that runs on the zIIP server are displayed in pink. If the server is running on a zAAP server, entries that are related to work that runs on the zAAP server are displayed in turquoise.

Procedure

1. From the Primary Option panel, enter B on the Option line.

   The Server Trace panel displays the most recent entries, which are at the end of the list. By default, the time, host name, and description of the event are displayed.
2. On the Server Trace panel, you can navigate through the trace messages in the following ways:
   - Use the UP, DOWN, RIGHT, and LEFT scroll commands (or their PF key equivalents) to navigate this panel.
   - Use the MAX or M scroll operand to scroll the maximum amount in any direction.
   - If you are at the beginning or end of the trace list (and it is full), press **ENTER** to scroll the list down. Messages are removed from the beginning and added to the end.
3. Optional: Perform any of the following steps:
   - To refresh the list, press Enter.
   - If you reposition the display, to see the most recent entries, issue the DOWN MAX command and then press **Enter**.
   - To display a different set of columns, type D on the command line, followed by the names of the columns to display.

Server Trace panel columns
Use the DISPLAY command to display specific columns on the Server Trace panel.

Table 38. Server Trace panel columns

Column	Description
ACTION	Displays one of the following: • ACC (accept) • REJ (reject) • NOA (no action)
ADDRESS	The location in memory of the actual record.
ADDRJOB	The location in memory of the current record in the JOBNAME vector.
ADDRUSR	The location in memory of the current record in the USERID vector.
APMRC	The APPC/MVS return code.
ASID	The address space ID of the user who created the current record.
AVZFLAGS	The bits that are set by the routines that created the trace.
CLOCK	The timestamp of when the record was created.
CNID	The identifier assigned to each thread that is created.
CODE	The lowest level return code for each event.
COLOR	The color assigned to a Server Trace message.
COUNT	The number of rules that processed the event.

Table 38. Server Trace panel columns (continued)

Column	Description
CPUTIME	The CPU time used by a particular thread. The format depends on how much CPU time the user has used: • Fewer than 1000 seconds: *nnn.nnns* • Between 1000 seconds and 100 hours: *hh:mm:ss* • 100 hours or more: *hhhh: mm*
CVID	The conversation ID that LU 6.2 assigns when a conversation starts.
DATE	The date when the message was created, in *dd:mm:yy* format.
ELAPSED	The total time that the current event used, in decimal microseconds (millionths of a second). To derive the total, the STCK (clock store) value that is taken at the beginning of processing is subtracted from the STCK value that is taken at the end of processing.
EVENT	The type of event that created the entry.
GTRIDTKN	The global transaction.
HOSTNAME	The TCP/IP host name or LU 6.2 host name.
HOSTX	The TCP/IP host name extended or the lU6.2 host name/mode.
IPADDR	The IP address, which is the TCP/IP source or target that is associated with the entry.
IPV6ADDR	Internet Protocol Version 6 address.
JOBNAME	The name of the job or address space that created the entry.
LENGTH	The length of the text section of the message.
LUNAME	The LU 6.2 source or target that is associated with the message.
MSGNO	The message number. When data collection begins, message 1 is the first message collected; message 2 is the second message; and so on. When there is no more room in the message area, the oldest message is discarded to make room for a new message. Therefore, the first message in the list might not be message 1.
MSGORIGN	The SIS/XCF (Instrumentation Server XCF) member name where the message originated. A message origin has the following format: *SYSIDALS_SSIDSISID* where • *SYSID* is the system ID. • *ALS_SSID* is the Data Virtualization Manager subsystem ID. • *SISID* is the Instrumentation Server ID.

Table 38. Server Trace panel columns (continued)

Column	Description
NODENAME	The name of the communications node that is associated with the message. The format of each entry depends on the communication link type.
OERC	The TCP/IP return code of the OE socket.
PATHID	IUCV path ID
PROCESS	OE Process ID, if task is dubbed
RC	The highest level return code for the message.
REASON	The second-level return code for the message.
RULESET	The name of the first RULESET.RULE that processed an event on NONE.NONE.
SECONDS	The first four byes of the binary timestamp, which indicates when the message was created.
SESSION	The communications session that is associated with the message. The format of each entry depends on the type of communication link.
SOCKET	The socket number that is associated with the message. This column applies only to TCP/IP events.
SQLRC	The SQL return code.
SSID	The subsystem ID, for example, Db2, IMS, or CICS.
TCBADDR	The TCB (task control block) address field that contains the address of the TCB that created the message.
TERMNAME	The name of the terminal that is associated with the event.
TIME	The time that the message was created, in *hh:mm:ss* format.
TIMEX	The time that the message was created, calculated to the microsecond, in *hh:mm:ss.uuuuuu* format.
TRACE1	The trace data that is specific to the message.
USERID	The security product user ID that best identifies the message.
VCID	The unique virtual connection ID.
VERSION	The version of the product that generated the message.
VTAMRC	The VTAM® return code.
XIDTOKEN	The XA token ID.

Displaying information about SQL entries

Use the SDINFO, SDTRAC, and SDDATA commands to display information about the SQL that is associated with a selected entry.

Procedure

1. In the **Command** field, type one of the following commands:

Command	Description
SDINFO	Invokes the SQL Explain program that presents explanatory text for the SQLCODE that is associated with the specified entry.
SDTRAC	Invokes the SQL Trace program that traces all SQL events for the connection ID that is associated with the specified entry.
SDDATA	Invokes the SQL Data program that presents a formatted SQL Communications Area (SQLCA) control block for the specified entry.

2. Position the cursor on the entry for which you want information, and press Enter.

Displaying information about the Data Virtualization Manager server
The TCBADDRESS column of the server log specifies whether the Data Virtualization Manager configuration is running on a zAAP or zIIP server or in SRB mode.

Procedure

On the command line, enter `d tcb` to display the TCBADDRESS column.

The leftmost hexadecimal digit in the high-order byte of the TCB address specifies the mode:

- 20 indicates that the Data Virtualization Manager server is running on a zAAP server.
- 80 indicates that the Data Virtualization Manager server is running in SRB mode.
- C0 indicates that the Data Virtualization Manager server is running on a zIIP server in SRB mode.
- D0 indicates that the Data Virtualization Manager server is running on a zIIP server in SRB mode and the zIIP server is running at a different speed than a CP (Turbo mode).

Locating entries in the server log

Use the LOCATE command to position the display at an entry that contains a specific date, time, message number, or label.

Procedure

1. From the **Primary Option** panel, enter B on the Option line.
2. On the **Server Trace** panel, use the DISPLAY command to display the appropriate column.
   For example, enter `D date`.
3. Enter the LOCATE command, followed by the criteria.
   For example, to find an entry that has the time 21:51:58, enter `L 21:51:58`.

   To specify criteria, use the following formats:

Criteria	Format
Time	One of the following: • *hh* • *hh:mm* • *hh:mm:ss*

Criteria	Format
Date	One of the following: • *dmmm*, single-digit date and current month • *ddmmm*, date and current month • *ddmmyy*, date, month, and 2-digit year • *ddmmyyyy*, date, month, and 4-digit year
Message number	The specific message number
Label	The previously specified label that was added to an entry

Filtering log entries

To view a subset of the log entries, create a profile. In the profile, you specify the criteria to use to select entries to display, and you select the specific events to display. The profile that you create affects only how you view log entries. Other users can create their own profiles.

Procedure

1. From the **Primary Option** panel, enter B on the Option line.
2. On the **Server Trace** panel, type `PROFILE` (with no operands) on the command line.
3. On the **Trace Browse Profile** panel, enter criteria in one or more of the following fields. If you enter multiple criteria, the values are joined with the logical AND operator. If you enter multiple values for a criterion, the values are joined with the logical OR operator. You can enter up to four values for each criterion.

Table 39. Profile filtering criteria

Criterion	Description
JOBNAME	Limits entries to those that contain the specified value in the JOBNAME column. You can use an asterisk (*) as a wildcard character.
USERID	Limits entries to those that contain the specified value in the USERID column. You can use an asterisk (*) as a wildcard character.
CONNECT	Limits entries to those that contain the specified value in the CONNECT column.
VCID	Limits entries to those that contain the specified value in the VCID (virtual connection ID) column.
HOST NAME	Limits entries to those that contain the specified value in the HOST NAME column. You can use an asterisk (*) as a wildcard character.
TCB	Limits entries to those that contain the specified value in the TCB column.
SSID	Limits entries to those that contain the specified value in the SSID column. You can use an asterisk (*) as a wildcard character.
XIDTOKEN	Limits entries to those that contain the specified value in the XIDTOKEN (XA token ID) column.

Table 39. Profile filtering criteria (continued)

Criterion	Description
GTRIDTKN	Limits entries to those that contain a matching GTRIDTKN (global transaction ID).
CONVTKN	Limits entries to those that contain a matching CONVTKN (conversation token ID).
MSGORIGIN	Limits entries to those that contain a matching MSGORGIN (message origin). You can use an asterisk (*) as a wildcard character. Use the following format to enter the values: *SYSIDALS_SSIDSISID* where • *SYSID* is the system ID. • *ALS_SSID* is the subsystem ID. • *SISID* is the Instrumentation Server ID.

4. Enter Y or N to include or exclude the following specific types of events from the result set:

Table 40. Profile filtering events

Event	Description
ABN	Abend entries.
ADA	ADABAS entries.
APM	APPC/MVS entries.
ATH	Authorization entries.
BKR	ACI broker entries.
CMD	Command entries.
CPG	C program entries.
DET	Detach entries.
DIS	Disable entries.
ECI	CICS EXCI entries.
ENA	Enable entries.
EXC	Exception entries.
FIL	File entries.
GLV	Global variable entries.
IMS	IMS entries.
MFL	MicroFlow (MFL) entries.
MQS	MQ message entries.
OTC	IBM OE sockets TCP/IP entries.
OTM	IMS/OTMA entries.
PUB	IBM Data Virtualization Manager for z/OS Streams entries.

Table 40. Profile filtering events (continued)

Event	Description
RPC	RPC entries.
RRS	RRS entries.
RSF	RRSAF entries.
SIS	Instrumentation Server entries.
SQL	SQL entries.
SOM	Security Optimization Management entries.
SQM	SQM entries.
SSL	SSL entries.
STG	Storage alteration entries.
STR	System trace entries.
TOD	Time-of-day entries.
TSO	TSO entries.
TXT	Product initialization, termination, and general execution entries.
TYP	TYP entries.
WLM	Workload Manager entries.
WWW	WWW entries.
XCF	Coupling Facility entries.
XTX	Extended text entries.
ZSR	Services entries.
6.2	LLU 6.2 entries.

5. Press **Enter** to save the profile.

Labeling and locating specific log entries

To quickly locate significant entries in the server log, replace the message number of an entry with a label.

About this task

After you add labels to entries the trace log, use the LOCATE command to find the entries.

Procedure

1. From the **Primary Option** panel, enter B on the Option line.
2. On the **Server Trace** panel, use the DISPLAY command to display the relevant columns.
   For example, enter `DISPLAY msgno date`.
3. When you locate the entry to which you want to add a label, edit the MSGNO column and enter a label that consists of a period and up to seven alphabetic characters.
   For example, enter `.POINTA`.
4. Enter the LOCATE command, followed by the criteria.

   To specify criteria, use the following formats:

Criteria	Format
Time	One of the following: • *hh* • *hh:mm* • *hh:mm:ss*
Date	One of the following: • *dmmm*, single-digit date and current month • *ddmmm*, date and current month • *ddmmyy*, date, month, and 2-digit year • *ddmmyyyy*, date, month, and 4-digit year
Message number	The specific message number
Label	The previously specified label that was added to an entry

Finding character strings in the server log

Use the FIND and RFIND commands to find a specific character string in the server log. You can find a string in a specific column or in a range of columns.

Procedure

1. From the **Primary Option** panel, enter B on the Option line.
2. On the **Server Trace** panel, enter the FIND command to find the character string.

   To search for a string in the USERID, EVENT, or SSID column, use the following syntax:

   ```
   FIND column-name string prefix direction
   ```

   Where

   - *column-name* is USERID, EVENT, or SSID.
   - *string* is the search string.
   - *prefix* specifies that the search string is generic and specifies only the prefix characters. Specify this argument when you search EVENT or SSID columns.
   - *direction* specifies the next match to find. Specify FIRST (default), LAST, PREV, or NEXT.

   To search for the string in a range of columns, use the following syntax:

   ```
   FIND TEXT string  direction start-column end-column msgno
   ```

   Where

   - TEXT is an optional keyword that indicates that you are searching only the text of the entries.
   - *string* is the search string. If the search string contains blank spaces or is identical to a FIND keyword, enclose the string in quotation marks. Enter an asterisk (*) to use the search string from the previous FIND command.
   - *direction* specifies the next match to find. Specify FIRST (default), LAST, PREV, or NEXT.
   - *start-column* specifies the number of the first column for the search.
   - *end-column* specifies the number of the last column for the search.
   - *msgno* is the maximum number of entries to search. The default is 5000.

The following FIND command searches for the string AVS1234W from the first message, beginning at column 10 and ending at column 30, for 10,000 messages:

```
F 'AVS1234W XYZ' 10 30 10000
```

3. Optional: Enter RFIND to repeat the previous FIND command.

Capturing the entries from the server trace

Use the P, PP, and SS commands to print server log entries to the ISPF list data set.

About this task

Each entry that you print contains the same columns that are displayed in the **Server Trace** panel and includes the entire contents of the text field. If the text field exceeds one line, the printed entry wraps to include three additional lines. Make sure that the ISPF list data set has enough space to hold the printed entries. The SS command requires more space than the PP command. The SS command prints 1 - 100 entries as they appear in the trace log, followed by the zoomed formatting for each entry, followed by the next 1 - 100 entries.

Procedure

On the **Server Trace** panel, to print log entries, perform one of the following steps:

- To print a single entry, enter P in the MESSAGENUM column.
- To print the summary information for a range of entries, enter PP in the MESSAGENUM column on the first and last entry in the range.
- To print the summary and detailed information, enter S in the MESSAGENUM column.
- To print the summary and detailed information for a range of entries, enter SS in the MESSAGENUM column on the first and last entry in the range.

Archiving the Server Trace

To save Server Trace information that is in virtual memory to a data set, enable Server Trace archiving.

About this task

Note: Do not enable the Trace Browse archive if you are using the Instrumentation Server, which handles archiving.

Procedure

To enable and configure Server Trace archiving, use the **`MODIFY PARM`** command and set the following parameters in the Data Virtualization Manager configuration member, AVZSIN00:

```
    “MODIFY PARM NAME(ARCHIVEDSNPREFIX) VALUE(HLQ.ARCHIVE)”
    “MODIFY PARM NAME(ARCHIVESTORCLASS) VALUE(SYSSMS)”
    “MODIFY PARM NAME(BROWSEARCHIVE) VALUE(AUTO)”
    “MODIFY PARM NAME(BROWSEARCHIVECOUNT) VALUE(30000)”
    “MODIFY PARM NAME(BROWSEARCHIVECUSHION) VALUE(15000)”
```

The following table lists the Server Trace archiving parameters:

Parameter	Description	Valid values
ARCHIVEDSNPREFIX	Defines the high-level qualifier, which the subsystem uses to construct data sets names for archive files. The value “.Dyyyyddd.Thhmmss” is appended to the qualifier, where yyyyddd is the Julian date, and hhmmss is the time of day.	‘NULL’
ARCHIVESTORCLASS	Defines the STORCLASS operand value that is used to define linear clusters for archive data sets. When not set, STORCLASS is not specified when the linear data sets are allocated.	‘NULL’
BROWSEARCHIVE	Controls whether the product produces archives of the wrap-around trace and how the archival procedure is inaugurated. Possible values are NONE, AUTO, and MESSAGE. When set to NONE, archival of the trace is not supported and only user-requested archive extracts are supported. Explicitly requested extract archives are not considered to be “backup” type archives. When set to AUTO, archival is triggered by automatically generating an ARCHIVE BACKUP command. When set to MESSAGE, a message is generated when reachieving should be performed. The generation of the ARCHIVE BACKUP command is not performed automatically.	AUTO MESSAGE NONE (default)
BROWSEARCHIVECOUNT	Specifies the number of messages to write for each automated archival operation. The archival process begins when the BROWSEARCHIVECOUNT value is reached. This value should be 30% of the BROWSEMAX parameter to allow the archival process to complete before the BROWSEMAX value is reached and trace browse records would be overwritten.	30000

Parameter	Description	Valid values
BROWSEARCHIVECUSHION	Specifies the number of messages that are used as a threshold for automated triggering of an archive event and as a guard against archiving overwritten messages. An archive event is scheduled for each group of BROWSEARCHIVECOUNT messages. However, scheduling is deferred until BROWSEARCHIVECUSHION additional messages have been logged. This cushion is required because some messages are updated in place, and allow the system to get beyond the ACTIVE message range before actually copying the messages to a backup. The cushion value is also used if a backup is requested and overlay of previously un-backed-up message is in progress or imminent. The system begins the archive with the next unarchived message, when possible. But if overlay is imminent or already in-progress, this number of messages is skipped in order to ensure that these overlaid messages are not copied. **Note:** The default setting for BROWSEARCHIVECUSHION is 50% of the BROWSEARCHIVECOUNT.	15000
BROWSEMAX	Specifies the maximum number of messages a trace holds. The maximum value allowed is 1910786.	100,000

System Management Facility logging

Using the System Management Facility (SMF), you can record system resource usage information in SMF data sets.

To enable SMF support during product customization, provide a value for the **SMF record number** product parameter. SMF logging can be used together with IBM Data Virtualization Manager for z/OS logging, or separately.

The following sections include SMF record subtype information.

Enabling SMF logging

Enable the SMF logging feature to collect and record information that is used to evaluate system usage. It can be used together with IBM Data Virtualization Manager for z/OS logging, or separately.

Procedure

Use the `MODIFY PARM` command to set the following parameter in the hlq.SAVZEXEC(AVZSIN00) member `IN00`:

```
"MODIFY PARM NAME(SMFNUMBER) VALUE(nnn)"
```

Where SMFNUMBER controls SMF recording. To enable SMF recording, set the value to the appropriate number for each SMF record subtype. If the value is set to zero, no logging takes place.

Results

After you enable SMF record subtypes, you can configure SMF parameter settings.

The following table lists and describes SMF parameters.

Table 41. SMF Parameters

Parameter	Description	Valid values
ADABASDBIDSMF	Causes one SMF record to be written per DBID accessed at the end of each session. The records contain command usage statistics. SMF Subtype 17: ADABAS Command by DBID Records	**YES** **NO** This is the default value.
CHECKSTORAGEINTERV	Controls how often (in seconds) statistics for allocated storage are gathered from the Data Virtualization Manager server. A value of zero turns off this function.	0
LOGERRORSSMF	Controls whether DB2 SQL error information should be written to SMF. When set to YES, this value generates SMF subtype 13 records. SMF Subtype 13: DB2 SQL Errors	**YES** **NO** This is the default value.
LOGINTERVALS	Controls whether session interval information is logged. Session interval information is logged by inserting rows in to a DB2 table. One row is inserted for each session at the end of each recording interval and at session termination time.	**YES** This is the default value. **NO**
LOGINTERVALSSMF	Controls whether session interval information should be written to SMF.	**YES** This is the default value. **NO**

Table 41. SMF Parameters (continued)

Parameter	Description	Valid values
LOGLSESSIONINTVALSMF	Controls whether interval type records are written to SMF. Interval records can also be written to the session log. SMF Subtype 02: Interval Summary Records	**YES** This is the default value. **NO**
LOGSTORAGESMF	Controls whether storage usage information should be written to SMF. Storage usage information can also be written to a DB2 table. SMF Subtype 09: Storage Interval Summary Records	**YES** **NO** This is the default value.
LOGWSTORTM	Enables logging Services information for Real-Time Monitoring. SMF Subtype 18: Services Records	**YES** **NO** This is the default value.
MONITORAPPC/MVS	Specifies whether to monitor APPC/MVS conversations. SMF Subtype 10: APPC/MVS Interval Summary Records	**YES** This is the default value. **NO**
MONRESPONSETIME	Controls whether to monitor the client response time if application names are defined in the initialization EXEC by using the DEFINE RTMONAPP statement. When set to YES, monitoring of the client response time occurs if application names are defined. SMF Subtype 14: Client Response Time Records	**YES** **NO** This is the default value.
PUBLISHINTERVALSMF	Specifies whether to write SMF records for the Streams long running tasks. SMF Subtype 19: Streams Records	**YES** **NO** This is the default value.

Table 41. SMF Parameters (continued)

Parameter	**Description**	**Valid values**
RECORDINGINTERVAL	Controls how often interval summary and per-client SMF and/or SQL records are created. These records show what resources were used during the current recording interval. The interval value is specified in seconds and should be a factor of one hour. The value should divide evenly into 3600.	900 (default)
SMFNUMBER	Controls SMF recording. To enable SMF recording, set SMFNUMBER to the desired number. If set to zero, no logging takes place. This number must be a unique SMF record.	0 (default) no logging
SMFRULEDISABLE	Indicates whether this type of SMF record should be written. SMF Subtype 03: SEF Rule Disablement Records	**YES** **NO** This is the default value.
SMFTRANSACT	Controls the creation of SMF transaction records. Possible values are YES and NO. When set to YES, an SMF record is created for each inbound client request. When set to NO, no per-transaction records are created. SMF Subtype 06: Per Transaction SMF Records	**YES** **NO** This is the default value.
WSSMF	Causes SMF records to be written for each Services transaction. WSSMF is only required if ends of session records are desired. If you only want summary records, this parameter should be set to NO. SMF Subtype 18: Services Records	**YES** **NO** This is the default value.

Table 41. SMF Parameters (continued)

Parameter	Description	Valid values
WSSMFSUMMARY	Causes summary SMF records to be written for Services transactions. Interval recordings of Services are summarized at the highest level (a single record per interval). Records have SM18RCTY = 'I'. SMF Subtype 18: Services Records	**YES** **NO** This is the default value.
WSSMFSUMMARYOPER	Causes Operation summary SMF records to be written for Services transactions. Enables interval recording of Services summarized at the operation level. Records have SM18RCTY = 'O'. SMF Subtype 18: Services Records	**YES** **NO** This is the default value.
WSSMFSUMMARYVDIR	Causes Virtual Directory summary SMF records to be written for Services transactions. Enables interval recording of Services summarized at the Virtual Directory level. Records have SM18RCTY = 'V'. SMF Subtype 18: Services Records	**YES** **NO** This is the default value.
WSSMFSUMMARYWS	Causes Virtual Directory summary SMF records to be written for Services transactions. Enables interval recording of Services summarized at the Web Service level. Records have SM18RCTY= 'W'. SMF Subtype 18: Services Records	**YES** **NO** This is the default value.

Record Subtype 01: Client System

This record is used to collect session and connection information about the client system.

About this task

The Subtype 01 record is used to record:

- End of session records, which indicate resource usage per client connection.
- Connection usage for a specific connection for the `INTERVAL RECORDING PERIOD` set by the `RECORDINGINTERVAL` parameter.

The information is written at the end of every connection.

Use the SM01RCTY field in the SMF record to set the record type to one of the following values:

- S: The final end-of-session record.
- F: The final interval record that shows the usage of CPU time for that specified interval.
- I: The interim interval record.

If you are only interested in end-of-session records for charge back situations, always check the SM01RCTY field for each 01 record to ensure that it is not an interval record; otherwise, incorrect calculations could be interpreted.

A sample SAS program is provided that can be used to print these SMF fields. The program is located in AVZSFDV1 in the hlq.SAVZCNTL member.

The following table lists the parameters used to configure the Subtype 01 record:

Table 42. Subtype 01 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X'10' = MVS/ESA 4 • X'08' = MVS/XA • X'04' = MVS/ESA • X'02' = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record sub type
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM01CLNA	CL16	Client system name
53	SM01CLTY	CL8	Client type (protocol type)
61	SM01CLUS	CL8	Client user id
69	SM01CLCP	D	Client CPU time
77	SM01SMID	CL4	Host system SMFID
81	SM01ODVR	XL1	ODBC version code
82	SM01ODRL	XL1	ODBC version code
83	SM01ODMD	XL2	ODBC modification code (MM/DD)
85	SM01ODYR	AL2	ODBC year value
87	SM01ODMN	AL1	ODBC month value
88	SM01ODDD	AL1	ODBC day value
89	SM01CNID	XL4	Connection ID
93	SM01LGTM	XL8	Client logon time (GMT TOD

Table 42. Subtype 01 Record Information (continued)

Offset	**Field Name**	**Field Subtype or Value**	**Description**
105	SM01ELTM	XL8	Client elapsed time (TOD)
113	SM01WRTO	XL8	Raw total bytes written
121	SM01TOTM	XL4	Client total response time
125	SM01HOTM	XL4	Client host response time
129	SM01ABCD	XL2	Client system abend code
131	SM01USAB	XL2	Client user abend code
133	SM01ENZQ	D	Enclave zIIP qualified CPU time
141	SM01ADLT	XL8	Adjusted client logon time
145	SM01IPAD	XL4	TCP/IP client IP address
153	SM01ORUS	CL8	Original user ID value
161	SM01PLAN	CL8	DB2 plan name
169	SM01SSNA	CL4	DB2 subsystem/group name
173	SM01DBCP	CL8	DB2 CPU time
181	SM01NTCP	CL8	Network CPU time
189	SM01OHCP	CL8	Other CPU time
197	SM01RXCP	CL8	REXX CPU time
205	SM01RPCP	CL8	RPC CPU time
213	SM01INST	CL8	Adjusted interval start time
221	SM01SQCN	F	SQL count
225	SM01SSAC	CL4	Group attachment member name
229	SM01ENCP	CL8	Enclave CPU time
238	SM01RCTY	C	Record type • C'F'=Final interval record type • C'I'=Interim interval record type • C'S'=Session record type
239	SM01APLN	H	Application name length
241	SM01APNA	CL18	Application name from client
261	SM01ENZI	D	Enclave zIIP CPU time
269	SM01ENZC	D	Enclave zIIP time on CP
277	SM01SLCP	D	SSL CPU time
291	SM01USLN	H	User parameter length
293	SM01USPA	CL100	User parameter from client
393	SM01PDSS	CL4	Product subsystem name
397	SM01CLWT	XL8	Client WAIT time

Table 42. Subtype 01 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
405	SM01CLRC	F	Client READ DATA COUNT
409	SM01LNID	CL100	Client LAN (network) user ID
509	SM01HONA	CL16	Host name (CMLI)
525	SM01ADCT	F	ADABAS command count
533	SM01SRCP	D	SRB CPU time

Record Subtype 02: Internal Summary

This record is used to collect session information for all users who are connected during a specific interval and the information is written at the end of each interval. All the resources that are used by all connections during that interval are recorded using this record.

About this task

The interval in which Subtype 02 records are written is determined by the `RECORDINGINTERVAL` parameter.

A sample SAS program is provided that can be used to print the fields in Subtype 02 records. The program is located in the `SMFSDB02` member of the hlq.SAVZEXEC(AVZSIN00) data set.

Interval summary records are automatically written if the `LOGINTERVALS` parameter is set to YES in the hlq.SAVZEXEC(AVZSIN00) member. You must have `LOGINTERVALS` enabled in order to also record Interval records into SMF.

Procedure

To log interval records to the logging tables but not log interval information to SMF, in the hlq.SAVZEXEC(AVZSIN00) member, set the `LOGINTERVALS` parameter as follows:

```
“MODIFY PARM NAME(LOGLSESSIONINTVALSMF) VALUE(NO)”
```

Where `LOGLSESSIONINTVALSMF` controls whether interval type records are written to SMF. Interval records can also be written to the session log.

Results

The following table lists the parameters used to configure the Subtype 02 record:

Table 43. Subtype 02 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)

Table 43. Subtype 02 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM02SMID	CL4	Host system (SMF ID)
41	SM02PDSS	CL4	Product subsystem name
45	SM02RCTY	C	Record type: C'I'=INTERVAL SUMMARY Record type
53	SM02INST	CL8	Interval start time
61	SM02SQCN	F	SQL COUNT
69	SM02ENCP	CL8	Enclave CPU time
77	SM02CLCP	CL8	Client task CPU time
85	SM02DBCP	CL8	DB2 CPU time
93	SM02NTCP	CL8	Network CPU time
101	SM02OHCP	CL8	OTHER CPU time
109	SM02RXCP	CL8	REXX CPU time
117	SM02RPCP	CL8	RPC CPU time
125	SM02ELTM	XL8	CLIENT ELAPSED time (TOD)
133	SM02WRTO	XL8	RAW TOTAL BYTES WRITTEN
141	SM02USCN	F	USER count FOR THIS INTERVAL
145	SM02MXUS	F	MAX INTERVAL CONCURRENT USERS
149	SM02RPHW	F	RPC HIGH WATER MARK
153	SM02RPCU	F	CURRENT NUMBER EXECUTING RPCS
157	SM02CLWT	XL8	CLIENT WAIT time
165	SM02CLRC	F	CLIENT READ DATA count
173	SM02ENZQ	D	Enclave zIIP QUALIFIED CPU time
181	SM02ENZI	D	Enclave zIIP CPU time
189	SM02ENZC	D	Enclave zIIP time ON CP
197	SM02SLCP	D	SSL CPU time
205	SM02SRCP	D	SRB CPU time

Record Subtype 03: SEF Rule Disablement

This record is created whenever an Event Facility (SEF) rule is disabled. All the resources that are used by all connections during that interval are recorded in this record.

About this task

These records are typically written when the Data Virtualization Manager server is shutdown. They are also written if a rule is manually disabled.

Procedure

To enable this record, use the **`MODIFY PARM`** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
"MODIFY PARM NAME(SMFRULEDISABLE) VALUE(YES)"
```

Where `SMFRULEDISABLE` indicates whether this type of SMF record should be written.

Results

The following table lists the parameters used to configure the Subtype 03 record:

Table 44. Subtype 03 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X'10' = MVS/ESA 4 • X'08' = MVS/XA • X'04' = MVS/ESA • X'02' = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM03RLTY	C	Rule type flag
38	SM03LACK	XL8	Last time this rule fired (TOD)
49	SM03PRCN	F	Process count
53	SM03FILI	F	Firing limit
57	SM03FIMX	F	Firing high water mark per interval
61	SM03RSNM	CL8	Ruleset name
69	SM03RLNM	CL8	Rule name
77	SM03ENTM	BL4	Rule enablement time (TIME BIN)

Table 44. Subtype 03 Record Information (continued)			
Offset	**Field Name**	**Field Subtype or Value**	**Description**
81	SM03ENDT	PL4	Rule enablement date (0CYYDDDF)
85	SM03CR	CL128	Rule criterion
213	SM03ENTT	XL4	Total enabled time in seconds

Record Subtype 04: Global Variable

This record is used to collect statistics about gloabl variable utilization.

About this task

No steps are required to enable the Subtype 04 record. A single Subtype 04 record is written by the Data Virtualization Manager server when it is shut down and the System Event Facility (SEF) is in use.

The following table lists the parameters used to configure the Subtype 04 record:

Table 45. Subtype 04 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X'10' = MVS/ESA 4 • X'08' = MVS/XA • X'04' = MVS/ESA • X'02' = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM04_OP_OFFSET	F	Offset to the permanent section
41	SM04_OP_LENGTH	H	Length of the permanent section
43	SM04_OP_NUMBER	H	Number of permanent sections
45	SM04_OT_OFFSET	F	Offset to the temporary section
49	SM04_OT_LENGTH	H	Length of the temporary section
51	SM04_OT_NUMBER	H	Number of temporary sections
53	SM04_OO_OFFSET	F	Offset to the opsvalue section
57	SM04_OO_LENGTH	H	Length of the OPSVALUE section
59	SM04_OO_NUMBER	H	Number of OPSVALUE sections
61	SM04_P_NUM_GLOBALS	F	Number of global variables (permanent section)

Table 45. Subtype 04 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
65	SM04_P_MAX_BLOCKS	F	Maximum number of blocks (permanent section)
69	SM04_P_HIGH_USED	F	High-used block count (permanent section)
73	SM04_P_IN_USE_BLKS	F	Number of in-use blocks (permanent section)
77	SM04_P_FREE_BLKS	F	Number of free blocks on free chain (permanent section)
81	SM04_P_FREE_AREAS	F	Number of free areas on free chain (permanent section)
85	SM04_P_PAGES	F	Number of pages in global workspace (permanent section)
89	SM04_P_UPDATES	F	Global variable update count (permanent section)
93	SM04_P_CHKPT_INTVL	F	SYSCHK1 checkpoint interval in seconds
97	SM04_P_CHKPT_COUNT	F	SYSCHK1 checkpoint count (permanent section)
101	SM04_P_CHKPT_RETRY	F	SYSCHK1 checkpoint retry count
105	SM04_P_ERRORS	F	Global variable error message count (permanent section)
109	SM04_T_NUM_GLOBALS	F	Number of global variables (temporary section)
113	SM04_T_MAX_BLOCKS	F	Maximum number of blocks (temporary section)
117	SM04_T_HIGH_USED	F	High-used block count (temporary section)
121	SM04_T_IN_USE_BLKS	F	Number of in-use blocks (temporary section)
125	SM04_T_FREE_BLKS	F	Number of free blocks on free chain (temporary section)
129	SM04_T_FREE_AREAS	F	Number of free areas on free chain (temporary section)
133	SM04_T_PAGES	F	Global variable update count (temporary section)
137	SM04_T_UPDATES	F	Global variable error message count (temporary section)
141	SM04_T_ERRORS	F	Global variable error message count (temporary section)
149	SM04_O_SYS_OPSVAL	F	Normal opsvalue function calls
153	SM04_O_GVAC_TOTAL	F	Internal OPSVALUE - unknown caller
157	SM04_O_GVAC_UNKNWN	F	Internal OPSVALUE - TOD catchup
161	SM04_O_GVAC_TODC	F	Internal OPSVALUE - TOD catchup
165	SM04_O_GVAC_EVENT	F	Internal OPSVALUE - GLVEVENT
169	SM04_O_JOBID	F	Internal OPSVALUE - GLVJOBID

Record Subtype 05: Services (Non-SOAP requests)

This record is used to log Services for non-SOAP Web requests.

About this task

The layout for the Subtype 05 record can be found in member `OPSMRC` of the *`hlq`*`.SAVZSAMP` data set.

A sample SAS program is provided which can be used to print these SMF fields. The program is located in member AVZSFDV5 of the hlq.SAVZCNTL member.

No steps are required to enable Subtype 05 records.

The following table lists the parameters used to configure the Subtype 05 record:

Table 46. Subtype 05 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X'10' = MVS/ESA 4 • X'08' = MVS/XA • X'04' = MVS/ESA • X'02' = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMF ID)
15	SMFHSSID	CL4	Subsystem ID (SWS_)
19	SMFHSUTY	BL2	Record subtype (05)
21	SMFHVRCD	CL8	SWS version code
29	SMFHRS00	CL8	Reserved for future use
37	SM05CLIP	CL16	Client IP address
53	SM05SMID	CL4	Host system SMFID
57	SM05PDSS	CL4	Product subsystem name
61	SM05CLUS	CL8	Client user ID or blanks
69	SM05AUTH	CL4	Client authorization status: NONE: Authorization not sent SENT: Authorization information sent but was not used by the server YES: Client user ID/password were valid NO: Client user ID/password were invalid
73	SM05RS00	CL4	Reserved for future use
77	SM05SRCP	D	CPU time that is used (TIMEUSED macro)
85	SM05CNID	XL4	Connection ID
89	SM05LGTM	XL8	Transaction connect time (GMT TOD)

Table 46. Subtype 05 Record Information (continued)

Offset	**Field Name**	**Field Subtype or Value**	**Description**
97	SM05ELTM	XL8	Transaction elapsed time
105	SM05WRTO	XL8	Total bytes written (raw)
113	SM05RS01	XL4	Reserved for future use
117	SM05ADLT	XL8	Transaction connect time (local TOD)
125	SM05MTCT	F	Count of URL matches processed
129	SM05ABCD	XL4	Transaction abend code (if any)
133	SM05ABRS	XL4	Transaction abend reason (if any)
137	SM05TRRC	F	Overall return code
141	SM05TRST	F	HTML status code
145	SM05TRRS	F	Reason code
149	SM05IPAD	F	IP address of client
153	SM05DBCP	CL8	DB2 CPU time (TOD Format)
161	SM05NTCP	CL8F	Network CPU time (TOD Format)
169	SM05RXCP	CL8	IBM Data Virtualization Manager for z/OS/REXX CPU time (TOD Format)
177	SM05RPCP	CL8	User program CPU time (TOD Format)
185	SM05OHCP	CL8	Other CPU time (TOD Format)
193	SM05SLCP	CL8	SSL processing CPU time (TOD Format)
201	SM05ENCP	CL8	Enclave CPU time (TOD Format)
209	SM05SRBT	CL8	SRB CPU time (TOD Format)
217	SM05RS02	CL8	Reserved for future use
225	SM05RDTO	XL8	Total bytes sent inbound
233	SM05INUR	CL128	Original inbound URL value
361	SM05RESC	F	Count of URL re-scans
365	SM0501CR	CL128	WWW rule criterion (URL match string)
493	SM0501RS	CL8	WWW rule event procedure set name
501	SM0501RL	CL8	WWW rule event procedure member name
509	SM0501EU	CL8	Runtime MVS user ID in effect (TOD Format)
517	SM05LSCR	CL128	WWW rule criterion (URL match string)
645	SM05LSRS	CL8	WWW rule event procedure set name
653	SM05LSRL	CL8	WWW rule event procedure member name
661	SM05LSEU	CL8	Runtime MVS user ID in effect (TOD Format)
669	SM05USR1	CL256	User data area 1
925	SM05USR2	CL256	User data area 2

Table 46. Subtype 05 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
1181	SM05ENZQ	D	Enclave zIIP qualified time (TOD Format)
1189	SM05ENZI	D	Enclave zIIP CPU time (TOD Format)
1197	SM05ENZC	D	Enclave zIIP time on CP (TOD Format)

Record Subtype 06: Per Transaction SMF Records

This record is used to log each inbound client request.

About this task

Each SMF transaction record contains information about all the work that is done on behalf of the client for each transaction request. The inbound client request may have caused zero, one, or more SQL operations to be run. A high number of Subtype 06 SMF records may be written in high volume environments because one SMF record is created for each transaction.

A sample SAS program is provided which can be used to print these SMF fields. The program is located in the hlq.SAVZEXEC(AVZSIN00) file data set.

Procedure

To enable this record, use the **`MODIFY PARM`** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
“MODIFY PARM NAME SMFTRANSACT VALUE(YES)”
```

Where SMFTRANSACT controls the creation of SMF transaction records. When set to YES, an SMF record is created for each inbound client request.

Results

The following table lists the parameters used to configure the Subtype 06 record:

Table 47. Subtype 06 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype

Table 47. Subtype 06 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM06CLNA	CL16	Client machine's hostname
53	SM06CLTY	CL8	Client communication type
61	SM06IPAD	XL4	IP address for TCP/IP clients
65	SM06CLUS	CL8	Client user ID
73	SM06CNID	XL4	Unique client connection ID
77	SM06SQOP	XL2	SQL operation code
79	SM06GNID	CL8	Generic user ID
87	SM06EXSZ	H	Extended user ID size
89	SM06EXID	CL50	Extended user ID area
89	SM06SIID	CL16	SQLESETI client user identification
105	SM06WSNA	CL18	SQLESETI client workstation name
139	SM06GNVL	CL1	Validation of generic ID
140	SM06SETI	CL1	Extended user ID IS SQLESETI Y or N
141	SM06PDSS	CL4	4-character IBM Data Virtualization Manager for z/OS subsystem name
145	SM06PLAN	CL8	DB2 plan name
153	SM06SSNA	CL4	DB2 subsystem name
157	SM06ADLT	XL8	Client logon time adjusted for GMT to local time
165	SM06ADCU	XL8	Current time (adjusted for GMT)
173	SM06ELTM	XL8	Elapsed time of the client connection
181	SM06SQEL	XL8	Current SQL statement elapsed time
189	SM06SQCP	XL8	Current SQL statement CPU time
197	SM06SQRC	F	Current SQL statement return code
201	SM06SQRE	F	Current SQL statement reason code
205	SM06SQSQ	F	Current SQL statement SQL CODE
209	SM06SQAB	F	Current SQL statement Abend code
217	SM06VCID	F	VCID of current user
221	SM06APPL	CL32	SQLESETI application name
221	SM06APNA	CL18	Application name
253	SM06ATKN	CL22	SQLESETI accounting token
281	SM06NASB	CL8	Natural subprogram name
289	SM06SQAC	F	Actual SQL string length

Table 47. Subtype 06 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
293	SM06SQLN	F	SQL source length
297	SM06SQSR	CL256	SQL source string

Record Subtype 09: Storage Interval Summary

This record is used to monitor Data Virtualization Manager server storage usage above and below the 16 MB threshold.

About this task

This record is written at the end of every Data Virtualization Manager server storage recording interval. They are set by the `CHECKSTORAGEINTERVAL` parameter. If the `CHECKSTORAGEINTERVAL` parameter is set to 0 (the default), storage usage recording in the Data Virtualization Manager server is disabled.

Procedure

To enable this record, use the **`MODIFY PARM`** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
“MODIFY PARM NAME(LOGSTORAGESMF) VALUE(YES)”
```

Where `LOGSTORAGESMF` controls whether storage usage information should be written to SMF. Storage usage information can also be written to a DB2 table.

Results

The following table lists the parameters used to configure the Subtype 09 record:

Table 48. Subtype 09 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM09SMID	CL4	Host system SMFID
41	SM09PDSS	CL4	Product subsystem name

Table 48. Subtype 09 Record Information (continued)			
Offset	**Field Name**	**Field Subtype or Value**	**Description**
45	SM09RCTY	C	Record type
53	SM09INST	CL8	Interval start time
77	SM09MXUS	F	Max interval concurrent user
81	SM09TSSP	F	Transient subpool
85	SM09TSBE	F	Transient HI ALLOC BTL
89	SM09TSAB	F	Transient HI ALLOC ATL
93	SM09HWBA	246D	HI ALLOC BTL HI ALLOC ATL

Record Subtype 10: APPC/MVS Interval Summary

This record is used to log APPC/MVS interval summary information.

Before you begin
APPC/MVS monitoring must be enabled for SMF recording of APPC/MVS summary records.

About this task
Subtype 10 records are written at the end of every Data Virtualization Manager server recording interval (which defaults to 15 minutes).

Procedure

To enable this record, use the **`MODIFY PARM`** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
“MODIFY PARM NAME(MONITORAPPC/MVS) VALUE(YES)”
```

Where `MONITORAPPC/MVS` specifies whether to monitor APPC/MVS conversations.

Results
The following table lists the parameters used to configure the Subtype 10 record:

Table 49. Subtype 10 Record Information			
Offset	**Field Name**	**Field Subtype or Value**	**Description**
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)

Table 49. Subtype 10 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM10SMID	CL4	Host system SMFID
41	SM10PDSS	CL4	Product subsystem name
45	SM10RCTY	C	Record type
53	SM10INST	XL8	Interval start time
77	SM10CVTO	F	Total conversations
81	SM10ALTO	F	Total allocated conversations
85	SM10SNTO	F	Total number of sends
93	SM10SDTO	D	Total data sent
101	SM10RCTO	F	Total number of receives
109	SM10RDTO	D	Total data received
117	SM10ACTO	F	Total active conversations

Record Subtype 11: APPC/MVS Conversation Summary SMF

This record is only used internally to display IMS APPC/MVS real-time detail.

About this task

No additional steps are required to enable Subtype 11 records.

The following table lists the parameters used to configure the Subtype 11 record:

Table 50. Subtype 11 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code

Table 50. Subtype 11 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
37	SM11SMID	CL4	Host system SMFID
41	SM11PDSS	CL4	Product subsystem name
45	SM11RCTY	C	Record type
53	SM11INST	XL8	Internal start time
77	SM11CVID	XL8	Conversation ID
85	SM11INOT	F	Inbound/Outbound indicator
89	SM11PLLO	F	Partner LU location
93	SM11CVKN	F	Conversation kind
97	SM11PLUW	XL26	Logical unit of work ID
123	SM11CVCO	XL8	Conversation correlator
131	SM11USID	CL10	Conversation Userid
141	SM11SCNM	CL8	Scheduler name
149	SM11TPNM	CL8	TP name
157	SM11LTPN	CL8	Local TP name
165	SM11LUNM	CL8	LU name
173	SM11PLNM	CL17	Partner LU name
193	SM11ARTM	XL8	Allocate arrival time
201	SM11AVTM	XL8	Conversation available time
209	SM11CSTM	XL8	Conversation start time
217	SM11CETM	XL8	Conversation end time
225	SM11MDNM	CL8	Mode name
233	SM11SYLV	F	Synchronization level
237	SM11SNTO	F	Total sends
245	SM11SDTO	D	Total data sent
253	SM11RCTO	F	Total receives
261	SM11RDTO	D	Total data received
269	SM11CSTO	F	Total callable service
273	SM11LSRC	F	Last service return code
277	SM11LSRE	F	Last service reason code
281	SM11CVST	F	Conversation state
285	SM11LSBT	XL8	Last service start time
293	SM11LSET	XL8	Last service end time
301	SM11URID	XL16	Unit of recovery identifier
317	SM11CNID	F	Connection ID

Table 50. Subtype 11 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
321	SM11CBAD	A	Count of URL re-scans

Record Subtype 13: DB2 SQL Errors

This record is used to record DB2 SQL errors.

About this task

This record is used for logging DB2 SQL errors. The LOGERRORSSMF parameter is used in addition to the LOGERRORS parameter, which logs DB2 SQL errors to a DB2 table.

Procedure

To enable this record, use the **MODIFY PARM** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
“MODIFY PARM NAME(LOGERRORSSMF) VALUE(YES)”
```

Where LOGERRORSSMF controls whether DB2 SQL error information should be written to SMF. Set the value to YES to generate SMF Subtype 13 records.

Results

The following table lists the parameters used to configure the Subtype 13 record:

Table 51. Subtype 13 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG • SMFHESA4 • SMFHXA • SMFHESA • SMFHVS2	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
9	SM13GNVL	CL1	VALIDATION OF GENERIC ID
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS VERSION CODE
37	SM13SMID	CL4	Host system SMFID
41	SM13PDSS	CL4	PRODUCT subsystem name
45	SM13RCTY	C	Record type
49	SM13SSAC	CL4	GROUP ATTACHMENT MEMBER name

Table 51. Subtype 13 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
69	SM13USID	CL8	CLIENT USER ID
77	SM13GNID	CL8	GENERIC USER ID
85	SM13EXID	CL(2+254)	EXTENDED USER ID
341	SM13HONA	CL(2+100)	CLIENT HOST name
441	SM13PRTY	CL(2+8)	PROTOCOL TYPE
453	SM13IPAD	XL4	IP ADDRESS FOR IP CLIENTS
457	SM13LUNA	CL(2+17)	LU name FOR LU 6.2 CLIENTS
477	SM13CNID	F	Session ID
481	SM13TMSP	CL8	CURRENT TIMESTAMP
489	SM13LGTM	CL8	LOGON TIMESTAMP
497	SM13APNA	CL(2+18)	APPLICATION name
517	SM13PLAN	CL8	DB2 plan name string
525	SM13SSNA	CL4	DB2 subsystem NAME STRING
529	SM13CUNM	F	Cursor number
533	SM13RC	F	Return code
537	SM13RECD	F	Reason code CODE
541	SM13SQCD	F	SQL CODE
545	SM13ABCD	F	ABEND CODE
549	SM13STNM	F	STATEMENT NUMBER
553	SM13STTY	F	STATEMENT TYPE

Record Subtype 14: Client Response Time

This record is used to capture client application response time exceptions.

About this task

The Subtype 14 record is written when a client application response time exception occurs. An exception occurs when the client measured response time is greater than the customer supplied response time for a particular application. Subtype 14 records are used with the Response Time Monitor feature.

Procedure

To enable this record, use the **MODIFY PARM** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
“MODIFY PARM NAME(MONRESPONSETIME) VALUE(YES)”
```

Where `MONRESPONSETIME` causes monitoring of the client response time if application names are defined in the Data Virtualization Manager configuration member by using the `DEFINE RTMONAPP` statement.

Results

The following table lists the parameters used to configure the Subtype 14 record:

Table 52. Subtype 14 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS VERSION CODE
37	SM14RCTY	C	Record type
41	SM14APNM	CL32	APPLICATION NAME
73	SM14LNID	CL100	CLIENT NETWORK USER ID
173	SM14IPAD	XL4	IP ADDRESS FOR IP CLIENTS
177	SM14USID	CL8	CLIENT USER ID
184	SM14DNDA	CL100	CLIENT DOMAIN NAME
285	SM14TMMI	F	Response time in milliseconds (This is the actual client response time for the transaction that produced the exception event)
289	SM14TRTR	F	Total number of client response time records
293	SM14SRTR	F	Sum of the total response time for all of the records
297	SM14TMGR	F	Total number of client response time records that missed the response time goal
301	SM14SMGR	F	Sum of the total response time for the records that missed the response time goal
305	SM14TGRT	F	Client response time goal (this is the acceptable response time)

Record Subtype 17: ADABAS Command by DBID Records

This record is used to capture the number of times a ADABAS database is accessed and the number of commands that were issued against the database before each session ended.

About this task

A Subtype 17 record is written for each Database ID (DBID) referenced and each record contains the number of times that commands were issued against the database before the session ended.

Procedure

To enable this record, use the **`MODIFY PARM`** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

```
"MODIFY PARM NAME(ADABASDBIDSMF) VALUE(YES)"
```

Where `ADABASDBIDSMF` causes one SMF record to be written per DBID accessed at the end of each session. The records contain command usage statistics.

Results

The following table lists the parameters used to configure the Subtype 17 record:

Table 53. Subtype 17 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZS)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM17SMID	CL4	Host system SMF identification
41	SM17PDSS	CL4	Product subsystem NAME
45	SM17ID	CL8	Connection ID
53	SM17LID	CL8	Logon user ID
61	SM17DBID	H	ADABAS identifier (DBID)
65	SM17A1	F	A1 COUNT
69	SM17BT	F	BT COUNT
73	SM17C1	F	C1 COUNT
77	SM17C3	F	C3 COUNT
81	SM17C5	F	C5 COUNT
85	SM17E1	F	E1 COUNT
89	SM17ET	F	ET COUNT
93	SM17HI	F	HI COUNT
97	SM17L1	F	L1 COUNT

Table 53. Subtype 17 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
101	SM17L4	F	L4 COUNT
105	SM17L2	F	L2 COUNT
109	SM17L5	F	L5 COUNT
113	SM17L3	F	L3 COUNT
117	SM17L6	F	L6 COUNT
121	SM17L9	F	L9 COUNT
125	SM17LF	F	LF COUNT
129	SM17N1	F	N1 COUNT
133	SM17N2	F	N2 COUNT
137	SM17RC	F	RC COUNT
141	SM17RE	F	RE COUNT
145	SM17RI	F	RI COUNT

Record Subtype 18: Services Records

This record is used to set the level of recording you want to use for SMF data for Services.

Table 54. Subtype 18 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (xDBy)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM18CLIP	CL16	Client IP address
53	SM18SMID	CL4	Host system SMFID
57	SM18PDSS	CL4	Product subsystem name
61	SM18CLUS	CL8	Client user ID or blanks

Table 54. Subtype 18 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
69	SM18AUTH	CL4	Client authorization status: • C'NONE' = None (authorization was not sent) • C'SENT' = Sent (authorization was sent) • C'YES' = YES (client user ID/password were valid) • C'NO' = NO (client user id/password were not valid)
73	SM18PORT	H	FMBIIG - Port number of session
75	SM18TYPE	C	Type of request: • C'W' = WEB SERVICE REQUEST • C'T' = TERMINAL SERVER REQUEST • C'C' = WSCICSCONN REQUEST
76	SM18RCTY	C	Record type: • C'S' = Session detail record type • C'I' = Interval summary record type • C'V' = Virtual directory summary • C'W' = Web service summary record • C'O' = Operation summary record
77	SM18SRCP	D	CPU time used TIMEUSED macro
85	SM18CNID	XL4	Connection ID
93	SM18LGTM	XL8	TRANS connect time (GMT TOD)
101	SM18ELTM	XL8	Transaction elapsed time
109	SM18WRTO	XL8	Total bytes written (RAW)
117	SM18ADLT	XL8	TRANS CONNECT TIME LOCAL TOD
125	SM18ABCD	XL4	Transaction abend code (if any)
129	SM18ABRS	XL4	Transaction abend reason (if any)
133	SM18TRRC	F	Overall return code
137	SM18TRST	F	HTML status code
141	SM18TRRS	F	Reason code
145	SM18IPAD	F	IP address of client
149	SM18DBCP	CL8	DB2 CPU time
157	SM18NTCP	CL8	Network CPU time
165	SM18RXCP	CL8	IBM Data Virtualization Manager for z/OS/REXX CPU time
173	SM18RPCP	CL8	User program CPU time
181	SM18OHCP	CL8	Other CPU time

Table 54. Subtype 18 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
189	SM18SLCP	CL8	SSL processing CPU time
197	SM18ENCP	CL8	Enclave CPU time
205	SM18RCCT	CL8	TRANS. Count for summary RCD
213	SN18SRBT	CL8	SRB CPU TIME
221	SM18RDTO	XL8	Total bytes sent in-bound
229	SM18INUR	CL128	Original in-bound URL value
357	SM18VDIR	CL128	Virtual directory
485	SM18WSNA	CL128	Web service
613	SM18NASP	CL128	Web service name space
741	SM18WSOP	CL50	Operation name
791	SM18WSTG	CL50	Target system name
841	SM18TRSE	C	SOAP fault length
842	SM18TRFX	CL256	SOAP fault text
1101	SM18ENZQ	D	Enclave zIIP qualified CPU time
1109	SM18ENZI	D	Enclave zIIP CPU time
1117	SM18ENZC	D	Enclave zIIP TIME on CP
1125	SM18INST	D	Adjusted interval start time

Record Subtype 18: Interval Usage Recording Options

This record is used to set the level of recording you want to use for SMF data for Services.

About this task

You can choose from four existing options to set the level of recording you want. The level is reflected in the Record Type Field (SM18RCTY) of the SMF record or RECORD_TYPE field in the DB2 record. You can choose different recording options for SMF and DB2, or you can choose to use one or all of the four recording options. When you choose more than one option, you get duplicate usage records summarized at different levels. Therefore, if the records are used for billing or usage information, care must be taken to not over calculate values that are based on the same usage information.

For example, resource usage for all the operations of a Web Service is reported in the Web Service level summary record, as well as in the operations summary records. If there is no Services activity at any of these levels, no record is written. The SM18RCTY field should be used to make this determination:

Procedure

1. To enable this record, use the **MODIFY PARM** command to set the parameter in the hlq.SAVZEXEC(AVZSIN00) member as follows:

   ```
   "MODIFY PARM NAME(LOGWSTORTM) VALUE(YES)"
   ```

   Where LOGWSTORTM enables logging Services information for Real-Time Monitoring.
2. Set the WSSMFSSUMMARY parameters as follows:

```
“MODIFY PARM NAME(WSSMFSUMMARY) VALUE(YES)”
“MODIFY PARM NAME(WSSMFSUMMARYOPER) VALUE(YES)”
“MODIFY PARM NAME(WSSMFSUMMARYVDIR) VALUE(YES)”
“MODIFY PARM NAME(WSSMFSUMMARYWS) VALUE(YES)”
```

Results

The following table lists the parameters used to configure the Subtype 18 record.

Table 55. Subtype 18 Record Information

Offset	**Field Name**	**Field Subtype or Value**	**Description**
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (The following table lists the parameters used to configure)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM19SMID	CL4	Host system SMFID
41	SM19PDSS	CL4	PRODUCT subsystem NAME
45	SM19RCTY	C	Record type: • C’I’ = INTERVAL Record type • C’F’ = FINAL Record type
46	SM19PBTY	C	SOURCE OR DESTINATION TYPE: • C’S’ = SOURCE TASK • C’D’ = DESTINATION TASK

Table 55. Subtype 18 Record Information (continued)

Offset	**Field Name**	**Field Subtype or Value**	**Description**
47	SM19TATY	C	SOURCE TYPE: • C'2' = DB2 SOURCE TASK • C'I' = IMS SOURCE TASK • C'C' = CICS SOURCE TASK • C'A' = ADABAS SOURCE TASK • C'D' = IDMS SOURCE TASK • C'V' = VSAM SOURCE TASK DESTINATION TYPE: • C'H' = HTTP DESTINATION • C'B' = MQ BROKER DESTINATION TASK • C'M' = MQ SERIES DESTINATION TASK
53	SM19INST	CL8	Interval start time
61	SM19TANA	CL8	Publish task name
69	SM19ENCP	CL8	Enclave CPU time
77	SM19CLCP	CL8	CPU time used
85	SM19DBCP	CL8	DB2 CPU time
93	SM19NTCP	CL8	Network CPU time
101	SM19OHCP	CL8	Other CPU time
109	SM19RXCP	CL8	IBM Data Virtualization Manager for z/OS/REXX CPU time
117	SM19RPCP	CL8	User program CPU time
125	SM19ELTM	XL8	Transaction elapsed time
133	SM19WRTO	XL8	Total bytes written
141	SM19SOCA	F	Number of events captured
145	SM19SOIG	F	Number of events ignored
149	SM19SORU	F	Number of rules run
153	SM19SOFA	F	Number of rule failures
157	SM19SOEQ	F	Number of events queued
165	SM19SOBC	D	Number of bytes captured
173	SM19SOBQ	D	Number of bytes queued
181	SM19DERD	F	Number of events read
185	SM19DESH	F	Number of events shipped
189	SM19DEBS	D	Number of bytes shipped
197	SM19DEFA	F	Number of records failed
201	SM19DEOP	F	Number of connection opens

Table 55. Subtype 18 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
205	SM19DERF	F	Number of retriable failures
213	SM19ENZQ	D	Enclave zIIP qualified CPU time
221	SM19ENZI	D	Enclave zIIP CPU time
229	SM19ENZC	D	Enclave zIIP time ON CP
237	SM19SLCP	D	SSL CPU time
245	SM19SRCP	D	SRB CPU time

Record Subtype 19: Streams

This record is used to write one row for each streams task, during each interval.

Procedure

To enable this record, set the following parameter in the hlq.SAVZEXEC(AVZSIN00) member.

```
“MODIFY PARM NAME(PUBLISHINTERVALSMF) VALUE(YES)”
```

Where PUBLISHINTERVALSMF controls whether to write SMF records for the Data Virtualization Manager server events long running tasks.

Results

The following table lists the parameters used to configure the Subtype 19 record:

Table 56. Subtype 19 Record Information

Offset	Field Name	Field Subtype or Value	Description
1	SMFHFG	BL1	Header flag byte: • X’10’ = MVS/ESA 4 • X’08’ = MVS/XA • X’04’ = MVS/ESA • X’02’ = VS2
2	SMFHRCTY	BL1	Record Type
3	SMFHTIME	BL4	Record written time (TIME BIN)
7	SMFHDATE	PL4	Record written date (0CYYDDDF)
11	SMFHSYID	CL4	System identification (SMFID)
15	SMFHSSID	CL4	Subsystem ID (AVZSIN00)
19	SMFHSUTY	BL2	Record subtype
21	SMFHVRCD	CL8	IBM Data Virtualization Manager for z/OS version code
37	SM19SMID	CL4	Host system SMFID
41	SM19PDSS	CL4	PRODUCT subsystem NAME

Table 56. Subtype 19 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
45	SM19RCTY	C	Record type: • C'I' = INTERVAL Record type • C'F' = FINAL Record type
46	SM19PBTY	C	SOURCE OR DESTINATION TYPE: • C'S' = SOURCE TASK • C'D' = DESTINATION TASK
47	SM19TATY	C	SOURCE TYPE: • C'2' = DB2 SOURCE TASK • C'I' = IMS SOURCE TASK • C'C' = CICS SOURCE TASK • C'A' = ADABAS SOURCE TASK • C'D' = IDMS SOURCE TASK • C'V' = VSAM SOURCE TASK DESTINATION TYPE: • C'H' = HTTP DESTINATION • C'B' = MQ BROKER DESTINATION TASK • C'M' = MQ SERIES DESTINATION TASK
53	SM19INST	CL8	Interval start time
61	SM19TANA	CL8	Publish task name
69	SM19ENCP	CL8	Enclave CPU time
77	SM19CLCP	CL8	CPU time used
85	SM19DBCP	CL8	DB2 CPU time
93	SM19NTCP	CL8	Network CPU time
101	SM19OHCP	CL8	Other CPU time
109	SM19RXCP	CL8	IBM Data Virtualization Manager for z/OS/REXX CPU time
117	SM19RPCP	CL8	User program CPU time
125	SM19ELTM	XL8	Transaction elapsed time
133	SM19WRTO	XL8	Total bytes written
141	SM19SOCA	F	Number of events captured
145	SM19SOIG	F	Number of events ignored
149	SM19SORU	F	Number of rules run
153	SM19SOFA	F	Number of rule failures
157	SM19SOEQ	F	Number of events queued
165	SM19SOBC	D	Number of bytes captured

Table 56. Subtype 19 Record Information (continued)

Offset	Field Name	Field Subtype or Value	Description
173	SM19SOBQ	D	Number of bytes queued
181	SM19DERD	F	Number of events read
185	SM19DESH	F	Number of events shipped
189	SM19DEBS	D	Number of bytes shipped
197	SM19DEFA	F	Number of records failed
201	SM19DEOP	F	Number of connection opens
205	SM19DERF	F	Number of retriable failures
213	SM19ENZQ	D	Enclave zIIP qualified CPU time
221	SM19ENZI	D	Enclave zIIP CPU time
229	SM19ENZC	D	Enclave zIIP time ON CP
237	SM19SLCP	D	SSL CPU time
245	SM19SRCP	D	SRB CPU time

DB2 logging

DB2 logging writes out the total z/OS resource usage information into a DB2 intervals table for a specified time interval.

The Data Virtualization Manager server also writes detailed information for each connection into a DB2 sessions table. When a client disconnects, a record is written to a DB2 sessions table. Use this information to provide detailed reporting of processor resource consumption in your client/server applications.

If SMF logging is also enabled, logging information is written to a set of DB2 tables, and extra Subtype 01 and Subtype 02 Records are written out to SMF. Sub Type 01 records are shared with normal end-of-sessions records. These records can be distinguished via the SM01RCTY field in the SMF type 01 record.

Enabling DB2 logging

Enables logging to DB2 tables using the Data Virtualization Manager servers.

Procedure

1. Use the **MODIFY PARM** command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(DEFAULTDB2PLAN) VALUE(SDBC1010)”
“MODIFY PARM NAME(DEFAULTDB2SUBSYS) VALUE(xxxx)”
“MODIFY PARM NAME(LOGDB2SUBSYS) VALUE(DSN)”
“MODIFY PARM NAME(LOGUSERID) VALUE(SDBB)”
“MODIFY PARM NAME(RECORDINGINTERVAL) VALUE(900)”
```

   The following table lists the parameters used to configure DB2 logging:

Parameter	Description	Valid values
DEFAULTDB2PLAN	Specifies the DB2 plan name that remote clients use to access DB2 when the connection is set to PLAN=DFLT. It is also used as the logging task's target DB2 subsystem when LOGDB2PLNAME is not specified. It is the plan that is used by Streams when connected to DB2.	SDBC1010
DEFAULTDB2SUBSYS	Specifies DB2 subsystem that remote clients use for access DB2 when the connection is set to SUBSYS=DFLT. It is also used as the logging task's target DB2 subsystem when LOGDB2SUBSYS is not specified.	'NONE'
LOGDB2SUBSYS	Controls the DB2 subsystem that is used for all SQL operations. If this parameter is set, then all logging operations are routed to the specified DB2 subsystem. If this parameter is not set, then each logging operation is routed to the DB2 subsystem that the operation was associated with or the default DB2 subsystem if the operation was not associated with any DB2 subsystem.	'NONE'
LOGUSERID	Controls the DB2 userid that is used for all SQL operations. This userid must have enough authority to update (insert) all of the tables modified by the logging task. If this field is not set, the main product address space userid is used for all update operations.	AVZS
RECORDINGINTERVAL	Controls how often interval summary and per-client SMF and/or SQL records are created. These records show what resources were used during the current recording interval. The interval value is specified in seconds and should be a factor of one hour. The value should divide evenly into 3600.	900

2. The logging tables are created by running the AVZD2LGT script, which is located in *hlq*.SAVZCNTL.
   The following tables are made available through logging:

- SQL
  - AVZ.APMVSSUM
  - AVZ.ERRORLOG
  - AVZ.INTERVALS
  - AVZ.SESSIONS
  - AVZ.SQLSOURCE
  - AVZ.STORAGE
- Services
  - AVZ.SERVICES
- Streams
  - AVZ.STREAMS

Record: Sessions

This record is used to log usage information for a specific connection during a specified time interval.

About this task

Sessions records get cut at client disconnect time, similar to when an SMF record is written for end of session records. Processor times are either for the entire session or for the interval, depending on the record type. The record type can be determined by the RECORD_TYPE field in the Session record. Values include:

- S: The final end-of-session record.
- F: The final interval record that shows the usage of processor time for that specified interval.
- I: The interim interval record.

Procedure

Use the **MODIFY PARM** command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
"MODIFY PARM NAME(LOGLSESSIONINTVALSMF) VALUE(YES)"
"MODIFY PARM NAME(LOGRETAINSESSIONS) VALUE(30)"
"MODIFY PARM NAME(LOGSESSIONS) VALUE(YES)"
"MODIFY PARM NAME(LOGSESSIONSTABLE) VALUE(DVS.SESSIONS)"
```

Results

The following table lists the parameters used to configure the Sessions record:

Table 57. Sessions Record for DB2

Parameter	Description	Valid values
LOGLSESSIONINTVALSMF	Controls whether interval type records are written to SMF. Interval records may also be written to the session log.	**YES** Default value is YES. **NO**

Table 57. Sessions Record for DB2 (continued)

Parameter	Description	Valid values
LOGRETAINSESSIONS	Controls the number of days to wait before automatically deleting rows from the sessions table. That is, all rows older than the number of days are deleted. If this value is zero, rows are never automatically deleted from the sessions table.	Number of days Default is 30 days.
LOGSESSIONS	Controls whether session information should be logged. Session information is logged by inserting rows in to a DB2 table. One row is inserted for each session at session termination time.	**YES** Default value is YES. **NO**
LOGSESSIONSTABLE	Sets the name of the DB2 table that is used to log session information. If a session is active, a row is inserted into this table as part of session termination.	DVS.SESSIONS Default name

AVZ.SESSIONS
The AVZ.SESSIONS table contains one Interval record for each SQL user for each recording interval. It also contains Sessions records that have the total information for the entire connection.

Column	Description
USERID	User ID associated with the record.
CLIENT_SYSTEM	Client PC name.
PROTOCOL	Either TCP/IP or LU 6.2.
RECORD_TYPE	Either Session or Interval.
TOTAL_CPUTIME	Total CPU time.
SRB_CPUTIME	SRB CPU time. Time is included in TOTAL_CPUTIME.
DATABASE_CPUTIME	Total database CPU time that is used by IMS and DB2. This field does not reflect CPU usage by RPCs that access IMS and DB2.
NETWORK_CPUTIME	Total network CPU time trappable within Data Virtualization Manager server. **Note:** Network CPU time cannot be trapped within Data Virtualization Manager server. Therefore, this field may not reflect the total CPU time.
REXX_CPUTIME	Total CPU time that is used by running SEF REXX programs.
RPC_CPUTIME	Total CPU time that is used by RPCs. This time includes CPU time that is used in DB2 and IMS if the RPCs accessed these databases.
SSL_CPUTIME	SSL processing CPU time.

Column	Description
ENCLAVE_CPUTIME	Total CPU time that is associated with the WLM enclave. **Note:** Other CPU times in this record may also be associated with the Enclave CPU time. Also, other tasks may have contributed CPU time to the same Enclave.
ZIIP_QUALIFIED	Enclave zIIP qualified CPU time.
ZIIP_CPU	Enclave zIIP CPU time.
ZIIP_ON_CP	Enclave zIIP time on CP.
OTHER_CPUTIME	All of the unaccountable CPU time.
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The 4-character IBM Data Virtualization Manager for z/OS subsystem ID.
DRIVER_VERSION	The IBM Data Virtualization Manager for z/OS Driver version.
DRIVER_DATE	The IBM Data Virtualization Manager for z/OS Driver date.
CONNECTION_ID	The unique Data Virtualization Manager server connection ID.
LOGON_TIME	Time the user logged on.
LOGOFF_TIME	Time the user logged off. For Interval records, this value is NULL.
INTERVAL_START	Interval start time. For Sessions records, this value is NULL.
CONNECT_TIME	Number of seconds the user was connected. For Interval records, this value is NULL.
BYTES_READ	Total number of bytes of data that is read from the client workstation.
BYTES_WRITTEN	Total number of bytes written to the client workstation.
COMMIT_COUNT	Total number of commits performed.
ROLLBACK_COUNT	Total number of rollbacks performed.
SQL_COUNT	Total number of SQL queries run.
RPC_COUNT	Total number of RPCs run.
ABEND_CODE	Abend for the session, if one occurred. For Interval records, this value is NULL.
IP_ADDRESS	For TCP/IP connections, this is the IP address of the client workstation; otherwise, this value is NULL.
LU_NAME	For LU 6.2 connections, the LU name that is used for the connection.
ORIGINAL_USERID	Original user ID, recorded in case it was changed by a SEF rule.
PLAN	DB2 plan that is used.
DATABASE	DB2 subsystem to which the user is connected.
DB_GROUP_MEMBER	DB2 group attachment member name.

Column	Description
APPLICATION	Application name. This value is set by the client’s ODBC connection information.
USERPARM	Optional **userparm** from the client. This value is set by the client’s ODBC connection information.
ELAPS_READ_TIME	The total number of read operations from Data Virtualization Manager server to the Client.
TOTAL_READ_COUNT	The total time (in seconds) that those reads were outstanding. If you ignore the transmission times, the time is the user response time.

Record: Interval

This record is used to log the total z/OS resources that are used by all connections from the starting interval time, until the next starting interval time.

About this task

This table contains one entry for each interval. The interval time frame is determined by the Data Virtualization Manager server RECORDINGINTERVAL parameter.

Interval records are written to SMF Subtype 02 records if the Data Virtualization Manager server is configured to write SMF records.

The following table lists the parameters used to configure the Interval record:

Procedure

1. To name the DB2 table that is to contain the interval, add the following parameter in the hlq.SAVZEXEC(AVZSIN00) member:

   ```
   “MODIFY PARM NAME(LOGINTERVALSTABLE) VALUE(AVZ.INTERVALS)”
   ```

2. Use the **MODIFY PARM** command in the hlq.SAVZEXEC(AVZSIN00) member, to enable or disable logging of SMF interval records.

   - To disable the logging of Interval records to SMF, set the LOGINTERVALSSMF parameter to NO:

     ```
     “MODIFY PARM NAME(LOGINTERVALSSMF) VALUE(NO)”
     ```

   - To enable the logging of Interval records to SMF, set the LOGINTERVALSSMF parameter to YES, and then set the LOGRETAININTERVALS parameter interval value accordingly:

     ```
     “MODIFY PARM NAME(LOGINTERVALS) VALUE(YES)”
     “MODIFY PARM NAME(LOGRETAININTERVALS) VALUE(30)”
     ```

Results

The following table lists the parameters used to configure the Interval Record.

Table 58. Interval Record for DB2

Parameter	Description	Valid values
LOGINTERVALS	Controls whether session interval information should be logged. Session Interval information is logged by inserting rows in to a DB2 table. One row is inserted for each session at the end of each recording interval and at session termination time.	**YES** Default value is YES. **NO**
LOGINTERVALSSMF	Controls whether session interval information should be written to SMF.	**YES** Default value is YES. **NO**
LOGINTERVALSTABLE	Specifies the name of the DB2 table that is used to log interval information. If interval recording is active, a row is inserted into this table at the end of each recording interval,	AVZ.INTERVALS Default name
LOGRETAININTERVALS	Controls the number of days to wait before automatically deleting rows from the interval summary table. That is, all rows older than the number of days are deleted. If this value is zero, rows are never automatically deleted from the interval summary table.	Number of days Default is 30 days.

AVZ.INTERVALS

The AVZ.INTERVALS table contains precise z/OS resource usage information for all SQL connections that were active for an interval. Each record in the table is associated with a starting interval time.

Column	Description
RECORD_TYPE	Describes the record type. Currently, this record type is always Summary.
TOTAL_CPUTIME	Total CPU time that is used by all connections.
SRB_CPUTIME	Time is included in TOTAL_CPUTIME.
DATABASE_CPUTIME	Total database CPU time that is used by all connections, currently consists of IMS and DB2. This field does not reflect CPU usage by RPCs that access IMS and DB2.
NETWORK_CPUTIME	Total network CPU time trappable within Data Virtualization Manager server. **Note:** Because some network CPU time cannot be trapped within Data Virtualization Manager server, this field may not reflect the total CPU time.
REXX_CPUTIME	Total CPU time that is used by running SEF REXX programs.

Column	Description
RPC_CPUTIME	Total CPU time that is used by RPCs. This includes CPU time that is used in DB2 and IMS if the RPCs accessed these databases.
SSL_CPUTIME	SSL processing CPU time.
ENCLAVE_CPUTIME	Total CPU time that is associated with the WLM enclave. **Note:** Other CPU times in this record may also be associated with the Enclave CPU time. Also, other tasks may have contributed CPU time to the same Enclave.
ZIIP_QUALIFIED	Enclave zIIP qualified CPU time.
ZIIP_CPU	Enclave zIIP CPU time.
ZIIP_ON_CP	Enclave zIIP time on CP.
OTHER_CPUTIME	All of the unaccountable CPU time that is associated with IBM Data Virtualization Manager for z/OS itself.
USER_COUNT	The number of users that were connected during this interval.
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The 4-character IBM Data Virtualization Manager for z/OS subsystem ID.
INTERVAL_START	Interval start time. For Sessions records, this value is null.
CONNECT_TIME	N/A. Set to null.
BYTES_READ	Total number of bytes sent from the client connections.
BYTES_WRITTEN	Total number of bytes of data that is written down to the client workstations.
COMMIT_COUNT	Total number of commits performed.
ROLLBACK_COUNT	Total number of rollbacks performed.
SQL_COUNT	Total number of SQL queries run.
RPC_COUNT	Total number of RPCs run.
MAXIMUM_USER	Maximum number of users this interval.

Record: SQL Source

This record is used to capture SQL information for use with the Server Activity Monitor (SAM).

About this task

This record can also be used with the Dynamic-to-Static Analyzer (DSA). Recording dynamic SQL statements in the table provides a central location for extracting dynamic SQL statements for input to the DSA application.

Procedure

Use the **`MODIFY PARM`** command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGRETAINSQL) VALUE(30)”
“MODIFY PARM NAME(LOGSOURCETABLE) VALUE(AVZ.SQLSOURCE)”
“MODIFY PARM NAME(LOGSQLSOURCE) VALUE(YES)”
```

The following table lists the parameters used to configure the SQL Source record:

Table 59. SQL Source Record for DB2

Parameter	Description	Valid values
LOGRETAINSQL	Specifies the number of days to wait before automatically deleting SQL from the SQL source table. That is, all rows older than the number of days are deleted. If this value is zero, then rows are never automatically deleted from the SQL source table.	Number of days Default is 30 days.
LOGSOURCETABLE	Sets the name of the DB2 table that is used to log SQL source for conversion from dynamic SQL to static SQL. Each SQL statement is stored in one or more rows of this table.	AVZ.SQLSOURCE Default name
LOGSQLSOURCE	Controls whether SQL source information should be logged. SQL source information is logged by inserting rows in to a DB2 table. When the SQL statement is processed, one row is inserted for each SQL statement. The logged SQL source is used to convert dynamic SQL to static SQL.	**YES** Default value is YES. **NO**

AVZ.SQLSOURCE
The AVZ.SQLSOURCE table contains all the dynamic SQL and CALL statements that are run by SQL.

Column	Description
USERID	User ID associated with the record.
GENERIC_ID	An alternative RACF/ACF2/Top Secret USER ID that an authorized Client (one serving multiple people) has passed stating that “this transaction is being run on behalf of this user.” This implementation is associated with Enterprise Auditing.
EXTENDED_USERID	User ID information that is passed by the Client and put into the SMF records.
CLIENT_SYSTEM	System machine name of client, for example, PC name or host name on UNIX.
IP_ADDRESS	For TCP/IP connections, this record is the IP address of the client workstation; otherwise, this value is NULL
LU_NAME	For LU 6.2 connections, the LU name that is used for the connection.
CONNECTION_ID	The unique Data Virtualization Manager server connection ID.

Column	Description
TIME_CURRENT	Timestamp for the current activity.
LOGON_TIME	Timestamp for the actual logon time.
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The 4-character IBM Data Virtualization Manager for z/OS subsystem ID.
APPLICATION	Application name. This value is set by the client's ODBC connection information.
PLAN	DB2 plan that is used.
DATABASE	DB2 subsystem to which the user is connected.
HASH_CODE	Internal Use.
CURSOR	Internal cursor name.
RETURN_CODE	Return code for non-DB2 users.
REASON_CODE	Code that is given as reason for a failure.
SQL_CODE	Return code for DB2 users.
ABEND_CODE	Abend for the mainframe session, if one occurred. For Interval records, this value is NULL.
TIMERONS	The DB2 estimated cost of the statement. A timeron is a unit of measurement used to give a rough relative estimate of the resources, or cost, required by the database server to execute two plans for the same query. The resources calculated in the estimate include weighted CPU and I/O costs.
SQL_LENGTH	The length of the SQL statement.
SQL	The actual SQL statement.
SQL_LOB	Object locator for large objects.
ROWID	DB2 row identifier.

Record: Storage

This record is used to monitor Data Virtualization Manager server storage usage.

About this task

The Storage record is written at the end of every Data Virtualization Manager server storage recording interval.

Procedure

To enable this record, use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
"MODIFY PARM NAME(CHECKSTORAGEINTERVAL) VALUE(900)"
"MODIFY PARM NAME(LOGSTORAGE) VALUE(YES)"
"MODIFY PARM NAME(LOGSTORAGESMF) VALUE(YES)"
"MODIFY PARM NAME(LOGSTORAGETABLE) VALUE(AVZ.STORAGE)"
```

The following table lists the parameters used to configure the Storage record:

Table 60. Storage Record for DB2

Parameter	Description	Valid values
CHECKSTORAGEINTERVAL	Controls how often (in seconds) statistics for allocated storage are gathered within IBM Data Virtualization Manager for z/OS. A value of zero turns off this function.	0 Default value of 0. Function is turned off.
LOGSTORAGE	Controls whether storage information is logged. Storage information is logged by inserting rows in to a DB2 table.	**YES** Default value is YES. **NO**
LOGSTORAGESMF	Controls whether storage usage information should be written to SMF. Storage usage information can also be written to a DB2 table.	**YES** **NO** Default value is NO.
LOGSTORAGETABLE	Sets the name of the DB2 table that is used to log storage information. A row is inserted into this table at the end of each recording interval, if storage logging is active.	AVZ.STORAGE Default name

AVZ.STORAGE
The AVZ.STORAGE is used to record the private and virtual storage that is used by the Data Virtualization Manager server address space in 15-minute intervals.

Column	Description
PRODUCT_SUBSYSTEM	The 4-character IBM Data Virtualization Manager for z/OS subsystem ID.
INTERVAL_START	The start time of summary activity.
MAXIMUM_USERS	The maximum number of users allowed.
SUBPOOL	Name of virtual storage information.
BELOW_16M	Amount of memory in use below 16 MB.
ABOVE_16M	Amount of memory is use above 16 MB.
SMFID	The SMFID as defined within Data Virtualization Manager server.

Record: APPC/MVS

This record is used to log APPC/MVS interval summary information by inserting rows in to a DB2 table. One row is inserted at the end of each recording level.

Procedure

To enable this record, use the **MODIFY PARM** command to add the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGAPMVSSUM) VALUE(YES)”
“MODIFY PARM NAME(LOGAPMVSSUMSMF) VALUE(YES)”
```

```
“MODIFY PARM NAME(LOGAPMVSSUMTABLE) VALUE(DVS.APMVSSUM)”
“MODIFY PARM NAME(LOGRETAINAPMVSSUM)VALUE(30)”
“MODIFY PARM NAME(MONITORAPPC/MVS) VALUE(YES)”
```

The following table lists the parameters used to configure the APPC/MVS record:

Table 61. APPC/MVS record for DB2

Parameter	Description	Valid values
LOGAPMVSSUM	Controls whether APPC/MVS interval summary information should be logged. APPC/MVS interval summary information is logged by inserting rows in to a DB2 table. One row is inserted at the end of each recording level.	**YES** **NO** Default value is NO.
LOGAPMVSSUMSMF	Controls whether APPC/MVS interval summary information should be written to SMF. APPC/MVS interval summary information can also be written to a DB2 table.	**YES** Default value is YES. **NO**
LOGAPMVSSUMTABLE	Sets the name of the DB2 table that is used to log APPC/MVS interval summary information. If APPC/MVS interval summary recording is active, a row is inserted into this table at the end of each recording interval.	AVZ.APMVSSUM Default name
LOGRETAINAPMVSSUM	Specifies the number of days to wait before automatically deleting rows from the APPC/MVS summary table. That is, all rows older than the number of days are deleted. If this value is zero, then rows are never automatically deleted from the APPC/MVS summary table.	0 Default value of 0. Function is turned off.
MONITORAPPC/MVS	Specifies whether to monitor APPC/MVS conversations. This parameter should be set to YES to activate the monitor.	**YES** Default value is YES. **NO**

AVZ.APMVSSUM
The AVZ.APMVSSUM table shows the APPC/MVS interval summary information.

Column	Description
RECORD_TYPE	Either Session or Interval.
TOTAL_CONV	The total number of conversations between TCP/IP and IBM Data Virtualization Manager for z/OS.
ALLOCATED_CONV	The number of conversations that are allocated to TCP/IP and IBM Data Virtualization Manager for z/OS.
NUMBER_OF_SENDS	The number of blocks of data sent.

Column	Description
DATA_SENT	The number of bytes of data sent.
NUMBER_OF_RECEIVES	The number of blocks of data received.
DATA_RECEIVED	The number of bytes of data received.
ACTIVE_CONV	The number of conversations in use at the end of the interval.
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The 4-character IBM Data Virtualization Manager for z/OS subsystem ID.
INTERVAL_START	Interval start time. For Sessions records, this value is null.

Records: Error Log

This record is used to log system errors.

Procedure

To enable this record, use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member.

```
“MODIFY PARM NAME(LOGERRORS) VALUE(YES)”
“MODIFY PARM NAME(LOGERRORSTABLE) VALUE(AVZ.ERRORLOG)”
“MODIFY PARM NAME(LOGRETAINERRORS) VALUE(30)”
```

The following table lists the parameters used to configure the Error Log record:

Table 62. Error Log records for DB2

Parameter	Description	Valid values
LOGERRORS	Controls whether error information should be logged. Error information is logged by inserting rows in to a DB2 table. When set to YES, one row is inserted for each error that is detected by the Data Virtualization Manager server address space or reported by an application running under the Data Virtualization Manager server address space.	**YES** Default value is YES. **NO**
LOGERRORSTABLE	Sets the name of the DB2 table that is used to log errors. A row is inserted into this table each time Data Virtualization Manager server detects an error. Errors can also be reported by applications running under the control of the Data Virtualization Manager server address space. **Note:** Error logging can be turned on and off at any time.	AVZ.ERRORLOG Default name

Table 62. Error Log records for DB2 (continued)

Parameter	Description	Valid values
LOGRETAINERRORS	Controls the number of days to wait before automatically deleting rows from the error logging table. For examples, all rows older than the number of days are deleted. If this value is zero, then rows are never automatically deleted from the error logging table.	30 Number of days. Default is 30.

AVZ.ERRORLOG
The AVZ.ERRORLOG table contains records of DB2 SQL failures that result from negative SQL return codes in SQL. Only DB2 is supported by this table.

Column	Description
USERID	User ID associated with the record.
GENERIC_ID	An alternative RACF/ACF2/Top Secret USER ID that an authorized Client (one serving multiple people) passed stating that “this transaction is being executed on behalf of this user”. This implementation is associated with Enterprise Auditing, formerly our Transaction Level Security (“TLS”).
EXTENDED_USERID	User ID information that is passed by the Client and put into the SMF records.
CLIENT_SYSTEM	System machine name of client, for example, PC name or host name on UNIX.
IP_ADDRESS	For TCP/IP connections, this record is the IP address of the client workstation; otherwise, this value is null
LU_NAME	For LU 6.2 connections, the LU name that is used for the connection.
CONNECTION_ID	The unique Data Virtualization Manager server connection ID.
TIME_CURRENT	Timestamp for the current activity.
LOGON_TIME	Timestamp for the actual logon time.
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The four-character IBM Data Virtualization Manager for z/OS subsystem ID.
APPLICATION	Application name. This value is set by the client’s ODBC connection information.
PLAN	DB2 plan that is used.
DATABASE	DB2 subsystem to which the user is connected.
DB_GROUP_MEMBER	DB2 group attachment member name.
CURSOR	Internal cursor name.
STATEMENT	The statement number.

Column	Description
TYPE	SQL.
RETURN_CODE	Return code for non-DB2 users.
REASON_CODE	Code that is given as reason for a failure.
SQL_CODE	Return code for DB2 users.
ABEND_CODE	Abend for the mainframe session, if one occurred. For Interval records, this value is NULL.
TIMERONS	The DB2 estimated cost of the statement.
SQLCA	SQL communication area, which provides an application program with information about the processing of SQL statements within the program.
SQL	The actual SQL statement.
MESSAGE	The message describing the error.

Record: Services

The Services records are used to set the level of recording you want to use.

About this task

There are existing recording level options from which you can choose. If you are using IBM Data Virtualization Manager for z/OS logging, the level is reflected in the `RECORD_TYPE` field in the DB2 record. If you are using SMF Records, the level is reflected in the **Record Type** field `SM18RCTY`.

You can choose to use one or all of the recording options. When you choose more than one option, you get duplicate usage records, summarized at different levels. Therefore, if the records are used for billing or usage information, care must be taken to not over calculate values that are based on the same usage information.

Procedure

1. To enable this record, use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member.

   ```
   “MODIFY PARM NAME (LOGWSTORTM) VALUE(YES)"
   ```

2. Optional: System page data sets should be reviewed and adjusted to hold more storage requirements.

   For each Web service run, 1664 bytes are needed in virtual storage. The length of time this information is retained is determined by the `WSMEMORYINTERVALS` parameter that defaults to `100`. The frequency of the recording of Summary records is determined by the `RECORDINGINTERVAL` parameter. These parameter settings, in addition to the number of Web service transactions run, affect the amount of storage required.

3. Select the type of recording interval that you want to use from the following level options:

   - Interval recording using no specific criteria.
   - Interval recording at the Virtual Directory level.
   - Interval recording at the Web Service level.
   - Interval recording at the operation level.
   - End of Session recording of each Web Service.

Results

The following table lists the parameters used to configure the Service record:

Table 63. Services Record for DB2

Parameter	Description	Valid values
LOGWSTORTM	Enables logging Services information for Real Time Monitoring (RTM).	**YES** **NO** Default value is NO.
RECORDINGINTERVAL	Controls how often interval summary and per-client SMF and/or SQL records are created. These records show what resources were used during the current recording interval. The interval value is specified in seconds and should be a factor of one hour. The value should divide evenly into 3600.	900 (default)
WSMEMORYINTERVALS	Sets the number of intervals to retain in memory for Services interval processing	100 (default)

Services interval recording using no specific criteria
This record enables logging interval information using no specific criteria.

Procedure

Use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGWSREQUESTSTABLE) VALUE(AVZ.SERVICES)”
“MODIFY PARM NAME(LOGRETAINWS) VALUE(30)”
```

The following table lists the parameters used to configure this record:

Table 64. Services Record for DB2: Interval recording using no specific criteria

Parameter	Description	Valid values
LOGRETAINWS	Controls the number of days to wait before automatically deleting rows from the Services table. That is, all rows older than the number of days are deleted. If this value is zero, then rows are never automatically deleted from the URLs table.	Number of days Default is 30 days.
LOGWSREQUESTSTABLE	Sets the name of the DB2 table that is used to log Services information. If a recording is active, a row is inserted into this table for each Web Service.	AVZ.SERVICES

Services interval recording at the Web Services level
This record enables logging interval information at the Web Services level.

Procedure

Use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member.

```
“MODIFY PARM NAME(LOGWSSUMMARY) VALUE(YES)”
“MODIFY PARM NAME(LOGWSSUMMARYWS) VALUE(YES)”
```

The following table lists the parameters used to configure the record:

Table 65. Services Record for DB2: Interval Recording at the Web Services Level

Parameter	Description	Valid values
LOGWSSUMMARY	Enables logging Services summary SMF records to a DB2 table. This parameter contains an overall summary of all Web Services requests, which are written at the end of a specified interval.	**YES** **NO** Default value is NO.
LOGWSSUMMARYWS	Enables logging Services Web Service summary SMF records to a DB2 table.	**YES** **NO** Default value is NO.

Services interval recording at the Web Services directory level
This record enables logging interval information at the Web Services directory level.

Procedure

Use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGWSSUMMARYVDIR) VALUE(YES)”
```

The following table lists the parameters used to configure this record:

Parameter	Description	Valid values
LOGWSSUMMARYVDIR	Enables logging Services Virtual Directory summary SMF records to a DB2 table.	**YES** **NO** Default value is NO.

Services interval recording at the operation level
This record enables logging interval information at the operation level.

Procedure

Use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGWSSUMMARYOPER) VALUE(YES)”
```

The following table lists the parameters used to configure for this record:

Parameter	Description	Valid values
LOGWSSUMMARYOPER	Enables logging Services Operation summary SMF records to a DB2 table.	**YES** **NO** Default value is NO.

Services End of Session recording for each web service
This record enables logging interval information for each web service.

Procedure

Use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGWSREQUESTS) VALUE(YES)”
```

The following table lists the parameters used to configure this record:

Table 66. Services for DB2: End of Session record

Parameter	Description	Valid values
LOGWSREQUESTS	Enables logging Services information. If recording is active, a row is inserted into a table for each Web Service. This may not be practical for large volumes of requests. Consider logging WS Summary records. **Note:** LOGWSREQUESTS should be set to YES in order to see the URL and Namespace data.	**YES** **NO** Default value is NO.

AVZ.SERVICES
The AVZ.SERVICES table shows statistics information for Services.

Column	Description
ORIGINAL_URL	The URL where the Web Service is stored.
CLIENT_USERID	The user ID of the client that sent the request.
CLIENT_SYSTEM	The system that sent the request.
CLIENT_AUTH_USAGE	The authentication mechanism used.
REQUEST_TYPE	The type of service request: Web Service, Terminal Service, or WSCICSCONN request.
RECORD_TYPE	The type of record this is: Interval Summary, Virtual Directory Summary, Web Service Summary, Operation Summary, or Session Detail.
RECORD_COUNT	Transaction count for Summary records.
TOTAL_CPUTIME	Total CPU time used. CPU time may also be accounted for in the Enclave CPU time.
SRB_CPUTIME	SRB CPU time. Time is included in TOTAL_CPUTIME.

Column	Description
DATABASE_CPUTIME	Total database CPU time that is used by IMS and DB2. This field does not reflect CPU usage by RPCs that access IMS and DB2. CPU time may also be accounted for in the Enclave CPU time.
NETWORK_CPUTIME	Total network CPU time trappable within Data Virtualization Manager server. CPU time may also be accounted for in the Enclave CPU time. **Note:** Network CPU time cannot be trapped within Data Virtualization Manager server. Therefore, this field may not reflect the total CPU time.
REXX_CPUTIME	Total CPU time that is used by running SEF REXX programs. CPU time may also be accounted for in the Enclave CPU time.
USER_PGM_CPUTIME	Total CPU time that is used by RPCs. This time includes CPU time that is used in DB2 and IMS if the RPCs accessed these databases. CPU time may also be accounted for in the Enclave CPU time.
SSL_CPUTIME	Total CPU time that is used in processing SSL routines. CPU time may also be accounted for in the Enclave CPU time.
ENCLAVE_CPUTIME	Total CPU time that is associated with the WLM enclave. **Note:** Other CPU times in this record may also be associated with the Enclave CPU time. Also, other tasks may have contributed CPU time to the same Enclave.
ZIIP_QUALIFIED	Enclave zIIP qualified CPU time.
ZIIP_CPU	Enclave zIIP CPU time.
ZIIP_ON_CP	Enclave zIIP time on CP.
OTHER_CPUTIME	All of the unaccountable CPU time. CPU time may also be accounted for in the Enclave CPU time.
CONNECTION_ID	The unique Data Virtualization Manager server connection ID.
GMT_LOGON	Time the user logged on in Greenwich Meridian Time.
LOGON_TIME	Time the user logged on.
CONNECT_TIME	The number of seconds the user was connected.
INTERVAL_START	The adjusted interval start time.
BYTES_READ	Total bytes read.
BYTES_WRITTEN	Total bytes written.
HTTP_STATUS	The HTTP Status code that is returned for this request.
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The name of the IBM Data Virtualization Manager for z/OS subsystem ID.
SERVER_CODE	Overall return code for the transaction.
SERVER_REASON	Associated reason code, if any.

Column	Description
SERVER_ABEND	Abend code for the transaction, if one occurred.
IP_ADDRESS	For TCP/IP connections, this is the IP address of the client workstation; otherwise, this value is null.
PORT	TCP/IP port number for this connection.
VIRTUAL_DIRECTORY	Alternate name (50 characters max) used to uniquely identify a virtual directory.
WEB_SERVICE	The name of the Web Service invoked.
NAME_SPACE	The namespace of the Web Service invoked.
OPERATION	The Operation of the Web Service invoked.
TARGET_SYSTEM	The name of the Target System used to process this request.
SOAP_FAULT	Some of the text of the Soap Fault message generated, if any.

Record: Streams

This record is used to write one SMF records for each Streams task, during each interval.

About this task

A 'Final' row is written covering a partial interval when a Streams task terminates. These can be distinguished by the SM19RCTY field of the SMF record, or the RECORD_TYPE column of the table. Some columns only apply to source tasks, and some columns apply only to destination tasks, so the remaining columns are null in any given row.

Procedure

Use the `MODIFY PARM` command to set the following parameters in the hlq.SAVZEXEC(AVZSIN00) member:

```
“MODIFY PARM NAME(LOGPUBINTERVALSTABLE) VALUE(AVZ.STREAMS)”
“MODIFY PARM NAME(LOGPUBINTERVALS) VALUE(YES)”
“MODIFY PARM NAME(LOGRETAINPUB) VALUE(30)”
```

The following table lists the parameters used to configure the Streams record:

Table 67. Streams Record for DB2

Parameter	Description	Valid values
LOGPUBINTERVALS	Enables logging of Streams interval information. If interval recording is active, a row is inserted into a table at the end of each recording interval,	**YES** **NO** Default value is NO.
LOGPUBINTERVALSTABLE	Sets the name of the DB2 table that is used to log Streams interval information. If interval recording is active, a row is inserted into this table at the end of each recording interval.	’AVZ.STREAMS’

Table 67. Streams Record for DB2 (continued)

Parameter	Description	Valid values
LOGRETAINPUB	Controls the number of days to wait before automatically deleting rows from the Streams log table. That is, all rows older than the number of days are deleted. If this value is zero, then rows are never automatically deleted from the Streams log table.	Number of days. Default is 0.

AVZ.STREAMS
The AVZ.STREAMS is used for capturing statistics for Streams usage.

Column	Description
RECORD_TYPE	Record type, either Interval record or Final Interval record (possibly a partial interval).
PUBLISH_TYPE	Record is for a Source or a Destination.
TASK_TYPE	Task type, either DB2 source, IMS source, CICS source, Adabas source, IDMS source, VSAM source, HTTP destination, MQ destination, or MQ Broker destination.
TASK_NAME	The name that is given to the task.
TOTAL_CPUTIME	Total CPU time. CPU time may also be accounted for in the Enclave CPU time.
SRB_CPUTIME	SRB CPU time. Time is included in TOTAL_CPUTIME.
DATABASE_CPUTIME	Total database CPU time that is used by IMS and DB2 Some CPU time may also be accounted for in the Enclave CPU time.
NETWORK_CPUTIME	Total network CPU time trappable within Data Virtualization Manager server. CPU time may also be accounted for in the Enclave CPU time. **Note:** Network CPU time cannot be trapped within Data Virtualization Manager server. Therefore, this field may not reflect the total CPU time.
REXX_CPUTIME	Total CPU time that is used by running SEF REXX programs. CPU time may also be accounted for in the Enclave CPU time.
SSL_CPUTIME	SSL processing CPU time.
ENCLAVE_CPUTIME	Total CPU time that is associated with the WLM enclave. **Note:** Other CPU times in this record may also be associated with the Enclave CPU time. Also, other tasks may have contributed CPU time to the same Enclave.
ZIIP_QUALIFIED	Enclave zIIP qualified CPU time.
ZIIP_CPU	Enclave zIIP CPU time.
ZIIP_ON_CP	Enclave zIIP time on CP.
OTHER_CPUTIME	All of the unaccountable CPU time. CPU time may also be accounted for in the Enclave CPU time.

Column	Description
SMFID	The SMFID as defined within Data Virtualization Manager server.
PRODUCT_SUBSYSTEM	The name of the IBM Data Virtualization Manager for z/OS subsystem ID.
INTERVAL_START	Interval start time.
ELAPSED_TIME	The length of the interval.
BYTES_WRITTEN	The number of bytes written to TCP/IP or MQ.
SO_EVENTS_CAPTURED	The number of events the source task captured.
SO_EVENTS_IGNORED	The number of events the source task was told to ignore.
SO_RULES_RUN	The number of times a source task ran a rule.
SO_RULES_FAILED	The number of times a source task ran a rule that failed.
SO_EVENTS_QUEUED	The number of events queued to a destination task.
SO_BYTES_CAPTURED	The number of events the source task captured.
SO_BYTES_QUEUED	The number of bytes in internal format that a source task queued to a destination.
DE_EVENTS_READ	The number of events that a destination task read.
DE_EVENTS_SHIPPED	The number of events that a destination task successfully shipped.
DE_BYTES_SHIPPED	The number of bytes in internal format that a destination task successfully shipped.
DE_EVENTS_FAILED	The number of events that a destination task had permanent failures.
DE_CONNECTIONS	The number of connections that a destination task established.
DE_RETRIABLE_FAILS	The number of events that a destination task had re-triable failures.

Chapter 8. Monitoring

Data Virtualization Manager server provides powerful diagnostic tools that can record critical events for individual transactions. This information can be used to diagnose, debug, and correct problems.

Data Virtualization Manager server provides the following trace options:

- Server Trace
- Instrumentation Server (IS)
- Server Trace Archival Facility
- SQL Tracing

Server Trace

The Server Trace adds Data Virtualization Manager server trace records to a trace buffer maintained in virtual storage. When the session is finished, the trace records are automatically saved in a VSAM data set.

Trace records are written for the following actions:

- SQL operations
- IMS calls
- CICS calls
- Communication events (LU 6.2, TCP/IP, and messages)
- Thread attach and detach events
- Remote Procedure Call (RPC) events
- Message events
- Errors (abends)

A Remote Procedure Call (RPC) can add its own trace messages to the trace for diagnostic purposes.

Using Trace Browse, you can perform the following actions:

- Display formatted columns of information, such as user ID and time
- Use FIND and LOCATE commands to search for data or a specific time and date
- Use the DISPLAY command to display additional columns of information
- Use the STATUS command to display the Trace Browse status area

In general, the Server Trace can accommodate the complete record of all client/server processing for several days. However, using hierarchical storage management, you can maintain an unlimited history of data. The Server Trace data collection routines support collection of all the data required for auditing, capacity planning, and trend analysis of usage patterns. You can set security for the Server Trace filter functionality to prohibit viewing of sensitive data by a non-authorized user.

Instrumentation Server

Using the Instrumentation Server (IS), you can run multiple instances of the server in a sysplex and route trace information to a single repository so that you have a global view of all activity.

Server Trace Archival Facility

Use the Server Trace Archival Facility to back up, or archive, active trace information. The archive consists of a large block of virtual storage, which can be backed up by a data-in-virtual (DIV) linear data set. This block of virtual storage is sub-divided into the following parts:

- The status area occupies the first 4 KB page of the virtual storage and contains checkpoint information about the trace area and information about the most recent trace archive.
- Event blocks begin in the second 4 KB page of the virtual storage area. Each event block occupies 896 bytes of storage. Each server event is recorded in the next available slot, beginning with the first slot, continuing to the end of the event blocks, and wrapping around to the beginning of the event block.
- Vector tables each begin on a 4 KB page boundary, and are located after the event blocks in the trace storage. Each vector table contains index information that allows views of the trace to be filtered without searching through the entire virtual storage area occupied by each individual event block.

SQL Trace

The SQL Trace program provides details about all of the SQL statements that applications issue. The information that is displayed in the SQL Trace program is derived from the main SDB log by using connection IDs as the selection criterion.

When you select an active session, the SQL Trace displays the current information. To refresh the information, press `Enter`.

Monitoring Data Virtualization Manager client response time

Client response time is the time between when a query starts on the Data Virtualization Manager server and when data is returned to the application.

Procedure

1. Use the `MODIFY PARM` command to add the following parameter that is located in the `AVZSIN00` configuration member:

   ```
   “MODIFY PARM NAME(MONRESPONSETIME) VALUE(YES)”
   ```

   The following table lists the parameter for configuring response time monitoring:

Parameter	Description	Valid values
MONRESPONSETIME	Controls whether to monitor the client response time for applications that are defined by the RTMONAPP parameter in the `AVZSIN00` configuration file. SMF Subtype 14: Client Response Time Records	**YES** Monitoring occurs. **NO** Default value is NO.

2. For each application that you want to monitor, add the following DEFINE statement to the `AVZSIN00` configuration member:

   `"DEFINE RTMONAPP APPLICATION(`*`appname`*`)", "TIME(`*`time`*`)"`

   Where:

   - *appname* is the application name, internal name, or module name.
   - *time* is the response-time goal, in milliseconds.

3. Restart the Data Virtualization Manager server so that the changes take effect.

Monitoring Streams with Server Trace

You can turn on tracing in Server Trace, but typically you do not need to refer to it unless instructed to by Technical Support.

Procedure

Enable tracing in Server Trace by using the `MODIFY PARM` command to set the following parameters that are located in the Data Virtualization Manager configuration member, `AVZSIN00`:

```
    “MODIFY PARM NAME(TRACEPUBLISH) VALUE(YES)”
    “MODIFY PARM NAME(TRACEPUBLISHCAPTURE) VALUE(YES)”
    “MODIFY PARM NAME(TRACEPUBLISHDATA) VALUE(NO)”
    “MODIFY PARM NAME(TRACEPUBLISHEVENTIO) VALUE(NO)”
    “MODIFY PARM NAME(TRACEPUBLISHFLOW) VALUE(YES)”
    “MODIFY PARM NAME(TRACEPUBLISHWORKIO) VALUE(NO)”
```

The following table lists the parameters for configuring basic Trace Browse support:

Parameter	**Description**	**Valid values**
TRACEPUBLISH	Controls tracing of Streams servers.	**YES** (default) All calls are traced. **NO**
TRACEPUBLISHCAPTURE	Controls tracing of the Streams event capturing.	**YES** **NO** (default) Streams events tracing is disabled.
TRACEPUBLISHDATA	Controls whether the full publish data for publish events is traced.	**YES** **NO** (default) Full publish data is not traced..
TRACEPUBLISHEVENTIO	*(Non-DB2 users only)* Controls tracing of the Streams input/ output to its event capture files.	**YES** **NO** (default) Input/output events to the event capture database are not traced.
TRACEPUBLISHFLOW	Controls tracing of the Streams module flow.	**YES** (default) The module flow is traced. **NO**
TRACEPUBLISHWORKIO	Controls tracing of the Streams input/output to its work files.	**YES** **NO** (default) Input/output to the Streams work file is not traced.

Instrumentation Server

The Instrumentation Server presents a global view of the Trace Browse facility by running many servers in a single logical partition (LPAR) or across a sysplex, and routing Trace Browse information to one repository. This is accomplished using Cross System Coupling Facility (XCF) services.

When multiple servers run in a single LPAR, XCF converts the call to a z/OS cross memory call so that XCF is not used.

The SIS ERRORONLY server limits the trace data being sent to the Instrumentation Server to contain only 'significant' information. This means that the IS Trace Browse wraps less frequently and is more useful. You use the SIS ERRORONLY server when you only want to trace error messages or if you are running in a high-volume environment.

The Instrumentation Server is not recommended for a high-volume environment where large amounts of tracing is occurring. The Instrumentation Server address space increases CPU usage by your IBM Data Virtualization Manager for z/OS environment based on the number of messages being processed. CPU increases could range around .0003 seconds of CPU per Trace Browse message. However, this number is relative to the type and speed of the processor where the IBM Data Virtualization Manager for z/OS subsystems are installed and should not be used as a definitive calculation. The Instrumentation Server may also affect transaction response time relative to the number of messages that are generated per transaction and the extra CPU costs for those messages.

Instrumentation Server benefits

There are many benefits to using the Instrumentation Server:

- The Instrumentation Server decreases the virtual storage requirements for the main Data Virtualization Manager server address spaces servicing transaction requests by removing the storage requirements for Trace Browse.
- The Instrumentation Server provides a global view of tracing activity for all your Data Virtualization Manager server instances in a single z/OS LPAR or across a sysplex into a single Instrumentation Server.
- The Instrumentation Server provides a larger trace data set to be used than the main Data Virtualization Manager server address space. This also allows for more trace data to remain in the Trace Browse display.
- You may also want to have one IS per IBM Data Virtualization Manager for z/OS subsystem to take advantage of the preceding first and third bullets.

Reducing the amount of tracing

Before you implement the Instrumentation Server, you should review your current Data Virtualization Manager server tracing values and reduce unnecessary tracing parameters to minimize the rate that messages are sent to the Instrumentation Server.

Procedure

To reduce the amount of tracing for a Data Virtualization Manager server and still benefit from tracing, use these recommendations:

- If you are doing a new install using the sample Data Virtualization Manager configuration member, `AVZSIN00`, shipped with the Data Virtualization Manager server distribution, ensure that all trace parameters are disabled by setting each parameter to `NO`. If you have trace options that have are added to `AVZSIN00` from an existing installation and that are not included in the distributed sample `AVZSIN00`, then comment these out to use the default values.
- Consider disabling the following parameters by adding them to the Data Virtualization Manager configuration member, `AVZSIN00`:
  - `TRACEIBMOEEVENTS` disables all TCP/IP tracing events.

- TRACETEXTEVENTS disables various informational messages that are associated with other trace events. This includes more logon messages, DB2 thread token value messages, and DB2 DBRM tracing messages.
- TRACESQMEVENTS disables the IBM Data Virtualization Manager for z/OS logging tracing used for logging activity in a set of predefined logging tables.
- TRACERRSEVENTS disables detailed RRS messages when using either the DB2 RRSAF interface or RRS two-phase commit support with any supported interface.
- TRACERPCEVENTS disables the tracing of the IBM Data Virtualization Manager for z/OS ODBC CALL RPC APIs. If you disable default trace options, be aware that Technical Support may require certain disabled trace options to be re-enabled to assist with any reported issues.

Installing the Instrumentation Server

To use either the Instrumentation Server or Instrumentation Server ERRORONLY, create a separate address space.

About this task

The sole responsibility of this address space is to act as the Instrumentation Server manager. Do not use this separate IS address space to run any client transactions other than IBM Data Virtualization Manager for z/OS Administrator functions. Use job AVZGNSIS located in the *hlq*.SAVZCNTL data set to help in performing the following steps 1 and 2. To install IS and IS ERRORONLY, take the following steps:

Procedure

1. Create a new VSAM data set for the Instrumentation Server address space.

   Increase the size of the Trace Browse data set to a minimum of 1750 cylinders for a 3390 device. This enables up to one million lines of tracing to be maintained in a single viewable Trace Browse.
2. Use job AVZGNSIS located in the *hlq*.SAVZCNTL data set to copy the following data sets:

   ```
   hlq.ATH.SAVZEXEC
   hlq.CMD.SAVZEXEC
   hlq.EXC.SAVZEXEC
   hlq.GLV.SAVZEXEC
   hlq.PUB.SAVZEXEC
   hlq.RPC.SAVZEXEC
   hlq.SQL.SAVZEXEC
   hlq.TOD.SAVZEXEC
   ```
3. Create a new startup JCL procedure for the Instrumentation Server address space. The AVZS member of the *hlq*.SAVZCNTL library contains sample JCL procedures for running the main address space (started task) as an Instrumentation Server address space.

   Add the AVZS PROC to the SYS1.PROCLIB or to another procedure library. You can change the name of the procedure to reflect the Instrumentation Server, and you can change the **SSID** parameter in the startup procedure to reflect another valid subsystem name.
4. Define the new started task to the security product.
5. Depending on the communication protocol that you use, obtain or define a new TCP/IP port or VTAM application ID.
6. Create a new Data Virtualization Manager configuration member for the Instrumentation Server address space. The Data Virtualization Manager configuration member is a REXX program that is used to set product parameters. A sample Data Virtualization Manager configuration member, AVZSIN00, is shipped with the product and can be customized to configure your Instrumentation Server address space. If you use a IBM Data Virtualization Manager for z/OS subsystem name other than AVZS for Instrumentation Server, be sure to rename this member to include the first four characters of the subsystem name.

   The sample AVZSIN00 contains only the required parameters for the Instrumentation Server. Make the following changes to the SIS AVZSIN00 configuration member:

   a) Define the TCP/IP Port number to be used.

b) Set up the SEF data sets.

c) Set up the data set definitions.

d) Update the **BROWSEMAX** parameter to match the desired size of the retained Trace Browse. You need 175 cylinders per 100000 Trace Browse messages.

e) Define the Instrumentation Server parameters, as follows:

- Use the **MODIFY PARM** command to set the following parameter that is located in the Data Virtualization Manager configuration member, AVZSIN00:

```
"MODIFY PARM NAME(SIS/XCF) VALUE(YES)"
```

- For each Data Virtualization Manager server you connect to the Instrumentation Server, add the following DEFINE statement that is located in the AVZSIN00 configuration member:

```
DEFINE SISXCF
  "CONNID(SYS1AVZS)",
  "SISID(AVZS)",
  "SISXCFGRP(SISAVZS)",
  "ERRORONLY(YES)", (for SIS ERRORONLY server)
  "MANAGER(NO)"
```

f) Make the following changes to each Data Virtualization Manager configuration member, AVZSIN00 that is connected to and sending Trace Browse messages to the Instrumentation Server. Use the **MODIFY PARM** command to set the following parameter:

```
"MODIFY PARM NAME(SIS/XCF)VALUE(YES)"
```

Note: Only one Data Virtualization Manager server can send data on a particular SISID-SISXCFGRP pair to an Instrumentation Server manager. This combination must be unique in the sysplex. For the Instrumentation Server manager, the CONNID must be unique in that Data Virtualization Manager server.

7. Create a AVZSINEF member in the SYSEXEC concatenation, where AVZS refers to the new Instrumentation Server subsystem name you are creating. A sample AVZSINEF is included in the distributed *hlq*.SAVZEXEC data set.
8. Start the Instrumentation Server address space.
9. Recycle all the Only one IBM Data Virtualization Manager for z/OS Servers connected to the Instrumentation Server address space. After these are recycled, view the transmitted Trace Browse messages from any Only one IBM Data Virtualization Manager for z/OS Server ISPF application by specifying the Instrumentation Server subsystem name in the SIS SSID field on the Only one IBM Data Virtualization Manager for z/OS Server Primary Option Menu.

Using the Instrumentation Server in a sysplex

The Cross System Coupling Facility (XCF) handles communication between Logical Partitions (LPARs) in a sysplex and is only used when setting up the Instrumentation Server to connect IBM Data Virtualization Manager for z/OS Servers and the Instrumentation Server across members of a sysplex. Information such as workload, status, and data transmission can be passed through the coupling facility. The information sharing is constant and continuous, allowing the independent z/OS images to know detailed information about the status of all images in the sysplex.

About this task

To use this most effectively, size the coupling facility for the Instrumentation Server that is based on the number of expected messages that are processed by the coupling facility. The sizing depends the Instrumentation Server's ability to read messages out of the coupling facility as the other IBM Data Virtualization Manager for z/OS Servers send messages to the coupling facility. If using the Instrumentation Server in a sysplex environment, you might have to resize the coupling facility to handle the increase in traffic for the Data Virtualization Manager server trace messages.

Use the IBM Resource Measurement Facility (RMF) to monitor the number of rejected messages in the XCF reports. If the reports contain many rejected messages, increase the size of the buffers. For more information about SCF reports, see Parallel Sysplex performance: XCF performance considerations at Parallel Sysplex Performance: XCF Performance Considerations.

There are no required structure definitions for the IS coupling facility. The Instrumentation Server uses the XCF signaling services, which use the paths that are defined in the `COUPLEXX` member of the system PARMLIB. These can be CTCs or CF structures.

If the member of the sysplex that hosts the Instrumentation Server terminates, all the IBM Data Virtualization Manager for z/OS Servers that are clients of the Instrumentation Server begin tracing locally. After the member of the sysplex and the Instrumentation Server are restarted, the IBM Data Virtualization Manager for z/OS Servers begin transmitting Trace Browse messages. Trace Browse messages that were written locally are not sent to the Instrumentation Server.

Monitoring and managing RRS transactions

IBM Data Virtualization Manager for z/OS Enterprise Transactions is a licensed add-on component of the IBM Data Virtualization Manager for z/OS product suite. This component supports the monitoring and management tasks of RRS (Resource Recovery Services) transactions.

With the RRS monitor and control options, you can use the Data Virtualization Manager server RRS manager to view and manage all in-progress two-phase commit protocol transactions that are managed by IBM Data Virtualization Manager for z/OS.

Note: IBM Data Virtualization Manager for z/OS Enterprise Transaction was designed and written to the Open Group XA-Distributed Transaction Protocol specification.

RRS Manager display

The Resource Manager program provides information about the Data Virtualization Manager server RRS Resource Manager. RRS is a z/OS component that uses two-phase commit protocol to manage transaction processing across multiple data sources. When two-phase commit support is enabled, IBM Data Virtualization Manager for z/OS registers its Resource Manager with RRS. The Resource Manager must be connected to RRS for two-phase commit transaction processing to occur.

Enabling two-phase commit transaction processing

When two-phase commit support is enabled, the Data Virtualization Manager Server registers its Resource Manager with RRS. The Resource Manager must be up and connected to RRS for two-phase commit transaction processing to occur.

About this task

To invoke RRS Manager information:

Procedure

1. From the Primary Option Menu, select **AVZ Admin** and press Enter.
2. From the **Server Management** menu, select **RRS** and press Enter.
3. From the **Server RRS Monitor** menu, select **Resource Manager** and press Enter.
4. Use the available line commands that are described in the following section to perform the appropriate functions.

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
D	Disables the Resource Manager.

Line commands	Description
E	Enables the Resource Manager.
F	Formats the information.
P	Prints the control block.
S	Displays the control block for the selected row.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description
RRS RESOURCE MANAGER NAME	The name of the RRS Resource Manger.
RRS STATUS	The status of the RRS Resource Manager program. Valid values are: • ACTIVE • DOWN • NO RRS (the RRS parameter is not been selected).
STATUS	The status of the RRS connection. • ENABLED allows normal transaction processing. • DISABLED means new RRS transactions are prevented from starting.
TRANSACTIONS STARTED	The number of transactions started successfully.
COMMITS NORMAL	The number of transactions committed normally.
COMMITS RECOVERY	The number of transactions committed by using the XA recover command.
COMMITS PANEL	The number of transactions committed manually.
ROLLBACKS NORMAL	The number of transactions rolled back normally.
ROLLBACKS RECOVERY	The number of transactions rolled back by using the XA recover command.
ROLLBACKS PANEL	The number of transactions rolled back manually.

Viewing active two-phase commit transactions

The Data Virtualization Manager server Active Transaction Control program allows you to view all active RRS transactions that are running in IBM Data Virtualization Manager for z/OS.

About this task

To invoke active RRS transaction control information:

Procedure

1. From the Primary Option Menu, select **AVZ Admin** and press Enter.
2. From the **Server Management** menu, select **RRS** and press Enter.
3. Select **Active Transactions** from the Data Virtualization Manager server **RRS Monitor** menu and press Enter.

Three panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

4. Use the available line commands that are described in the following section to perform the appropriate functions.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
F	Formats the information for the selected row.
P	Prints the control block for the selected row.
S	Displays the control block for the selected row.
T	Displays the Server Trace data that is related to this RRS XID.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
GTRID LENGTH	The length of the XA global transaction ID.	
GLOBAL TRAN ID	The first 32 bytes of the XA global transaction ID.	GTRID
TRXN BEGIN TIME	The date and time when the transaction began running.	START
CLIENT USERID	The user ID passed by the client.	USERID
RRS STATE	The state of the transaction according to RRS.	TMTYPE
TRANSACTION TYPE	The type of transaction manager on the client side that is coordinating this transaction: Tuxedo or MTS.	TMTYPE
NUMBER OF THREADS	The number of z/OS threads (1 - 8) participating in the transaction.	TMTYPE
RRS URID	The RRS-assigned Unit of Recovery (UR) ID for the first or only thread of this transaction. It can be used to correlate this transaction with an RRS UR.	URID
XID TOKEN	The assigned token that is associated with this transaction. It can be used with PROFILE and DISPLAY in the Server Trace Facility.	XTOKEN

Column name	Description	Sort name
BQUAL LENGTH	The length of the XA branch qualifier.	XTOKEN
BQUAL VALUE	The first 32 bytes of the XA branch qualifier value, up until the last valid byte.	XTOKEN
GTRID 2ND HALF	The second 32 bytes of the XA global transaction ID, up until the last valid byte.	XTOKEN

Viewing indoubt two-phase commit transactions

The Indoubt Transaction program displays RRS transactions that are in the indoubt state and allows the user to commit or rollback these transactions.

About this task

⚠ **Warning:** The RRS transactions that run under IBM Data Virtualization Manager for z/OS on z/OS can be one part of a larger transaction that is coordinated by the client side transaction manager (TUXEDO or MTS). Issuing a `COMMIT` or `ROLLBACK` could leave the overall transaction, as well as its data, in an inconsistent state. Extreme care must be used with these commands.

To invoke indoubt RRS transaction information:

Procedure

1. From the Primary Option Menu, select **AVZ Admin** and press Enter.
2. From the **Server Management** menu, select **RRS** and press Enter.
3. Select **Indoubt Transactions** from the **RRS Monitor** menu and press Enter.

   Three panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.
4. Use the available line commands that are described in the following section to perform the appropriate functions.

Available commands

This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
C	Commits the transaction (see warning).
F	Formats the information for the selected row.
P	Prints the control block for the selected row.
R	Commits the transaction (see warning).
S	Displays the control block for the selected row.
T	Displays the Server Trace data that is related to this RRS XID.

⚠ **Warning:** Using the C or R line commands can leave the overall client transaction and the data in an inconsistent state.

Column names

The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
GLOBAL TRAN ID	The XA global transaction ID assigned by the client-side transaction manager.	GTRID
TRXN BEGIN TIME	The date and time when the transaction began running on IBM Data Virtualization Manager for z/OS on the MVS system.	START
CLIENT USERID	The user ID passed by the client.	USERID
RRS STATE	The state of the transaction according to RRS.	USERID
TRANSACTION TYPE	The type of transaction manager on the client side that is coordinating this transaction: Tuxedo or MVS.	TMTYPE
NUMBER OF THREADS	The number of MVS (1 - 8) threads participating in the transaction.	TMTYPE
RRS URID	The RRS-assigned Unit of Recovery (UR) ID for the first or only thread of this transaction.	URID
XID TOKEN	The token that is associated with this transaction. It can be used with PROFILE and DISPLAY in the Server Trace Facility.	XTOKEN
BQUAL LENGTH	The length of the XA branch qualifier.	XTOKEN
BQUAL VALUE	The first 32 bytes of the XA branch qualifier value, up until the last valid byte.	XTOKEN
GTRID 2ND HALF	The second 32 bytes of the XA global transaction ID, up until the last valid byte.	XTOKEN

Displaying information about failed two-phase commit transactions

The Recovery Table program displays RRS transactions that are stored in the RRS recovery table because of a failure while the transaction was in progress.

About this task

To invoke the recovery table display:

Procedure

1. From the Primary Option Menu, select **AVZ Admin** and press Enter.
2. From the **Server Management** menu, select **RRS** and press Enter.
3. Select **Recovery Table** from the **RRS Monitor** menu and press Enter.

Three panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.

4. Use the available line commands that are described in the following section to perform the appropriate functions.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
F	Formats the information for the selected row.
P	Prints the control block for the selected row.
S	Displays the control block for the selected row.
T	Displays the Trace Browse data that is related to this RRS XID.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
GLOBAL TRAN ID	The XA global transaction ID assigned by the client-side transaction manager.	GTRID
TRXN BEGIN TIME	The date and time when the transaction began running.	START
CLIENT USERID	The user ID passed by the client.	USERID
RRS STATE	The state of the transaction according to RRS.	TMTYPE
TRANSACTION TYPE	The type of transaction manager on the client side that is coordinating this transaction: Tuxedo or MTS.	TMTYPE
NUMBER OF THREADS	The number of MVS (1 - 8) threads participating in the transaction.	TMTYPE
RRS URID	The RRS-assigned Unit of Recovery (UR) ID for the first or only thread of this transaction.	URID
XID TOKEN	The token that is associated with this transaction. It can be used with PROFILE and DISPLAY in Server Trace.	XTOKEN
BQUAL LENGTH	The length of the XA branch qualifier.	XTOKEN
BQUAL VALUE	The first 32 bytes of the XA branch qualifier value, up until the last valid byte.	XTOKEN

Column name	Description	Sort name
GTRID 2ND HALF	The second 32 bytes of the XA global transaction ID, up until the last valid byte.	XTOKEN

Invoking the RRS Units of Recovery information

The Units of Recovery program displays the RRS Units of Recovery (URs) associated with this instance of the Data Virtualization Manager server.

About this task

To invoke the Units of Recovery program:

Procedure

1. From the Primary Option Menu, select **AVZ Admin** and press Enter.
2. From the Server Management Menu, select **RRS** and press Enter.
3. Select **Unit of Recovery** from the **RRS Monitor** menu and press Enter.

   Three panels comprise this program. Use the LEFT and RIGHT scroll commands (or PF keys) to shift between them.
4. Use the available line commands that are described in the next section to perform the appropriate functions.

Available commands
This program supports all four scrolling commands (UP, DOWN, LEFT, RIGHT) and their PF key equivalents or scroll bar equivalents.

It also supports the primary SORT and LOCATE commands and the following line commands:

Line commands	Description
F	Formats the information for the selected row.
P	Prints the control block for the selected row.
S	Displays the control block for the selected row.
T	Displays the Server Trace data that is related to this RRS XID.

Column names
The following table describes each column name on the ISPF panels and provides a sort name (if available).

Column name	Description	Sort name
GLOBAL TRAN ID	The XA global transaction ID assigned by the client-side transaction manager.	GTRID
TRXN BEGIN TIME	The date and time when the transaction began.	START
CLIENT USERID	The user ID passed by the client.	USERID
RRS STATE	The state of the transaction according to RRS.	USERID

Column name	Description	Sort name
TRANSACTION TYPE	The type of transaction manager on the client side that is coordinating this transaction: Tuxedo or MTS.	TMTYPE
NUMBER OF THREADS	The number of MVS (1 - 8) threads participating in the transaction.	TMTYPE
RRS URID	The RRS-assigned Unit of Recovery (UR) ID for the first or only thread of this transaction.	URID
XID TOKEN	The token that is associated with this transaction. It can be used with PROFILE and DISPLAY in Server Trace.	URID
BQUAL LENGTH	The length of the XA branch qualifier.	URID
BQUAL VALUE	The first 32 bytes of the XA branch qualifier value, up until the last valid byte.	URID
GTRID 2ND HALF	The second 32 bytes of the XA global transaction ID, up until the last valid byte.	URID
TCB ADDRESS	The initial TCB on which the UR ran. (Zero if the UR was from the recovery table.)	TCB
VCID	The VCID value that is assigned when this UR started.	VCID
PREPARE RET CODE	The return code from the RRS PREPARE operation. (N/A if the PREPARE operation is not done).	VCID
COMMIT RET CODE	The return code from the RRS COMMIT operation. (N/A if the RRS COMMIT operation is not done).	VCID
ROLLBACK RET CODE	The return code from the RRS ROLLBACK operation. (N/A if the RRS ROLLBACK operation is not done).	VCID
FORGET RET CODE	The return code from the RRS FORGET operation. (N/A if the RRS FORGET operation is not done).	VCID
TRANSACTION FLAG 1	Transaction flag 1. Used for diagnostics.	VCID
TRANSACTION FLAG 2	Transaction flag 2. Used for diagnostics.	VCID

Column name	Description	Sort name
TRANSACTION FLAG 3	Transaction flag 3. Used for diagnostics.	VCID
TRANSACTION FLAG 4	Transaction flag 4. Used for diagnostics.	VCID
TRANSACTION FLAG 5	Transaction flag 5. Used for diagnostics.	VCID
DIAGNOSTIC FLAG 1	Diagnostic flag 1. Used for diagnostics.	VCID
DIAGNOSTIC FLAG 2	Diagnostic flag 2. Used for diagnostics.	VCID

Chapter 9. Managing users and system resources

This chapter describes the methods that are used to streamline the management of system resources. These methods allow you to maintain response times in pre-established service levels as the numbers of users grow from a few to tens of thousands.

System resources management

IBM Data Virtualization Manager for z/OS provides several system resources that are used to streamline the management of Data Virtualization Manager server performance, helping to maintain response times within pre-established services levels as numbers of users grow from a few to tens of thousands.

These resources include:

- Block fetch
- CPU time limits
- Wait time for all clients
- Program execution duration time limit
- Session failures
- Dispatch priority

Enabling time limits

Data Virtualization Manager server provides an external error, failure, and warning time limits for all clients.

Procedure

To enable the external CPU time and external time limits, use the `MODIFY PARM` command to add the following parameters to the `AVZSIN00` configuration member:

```
if 1 = 1 then
  do
    “MODIFY PARM NAME(CHECKLIMITSINTERVAL) VALUE(15 SECONDS)”
    “MODIFY PARM NAME(ERRORCPUTIME) VALUE(0 SECONDS)”
    “MODIFY PARM NAME(ERRORWAITTIME) VALUE(0 SECONDS)”
    “MODIFY PARM NAME(FAILCPUTIME) VALUE(0 SECONDS)”
    “MODIFY PARM NAME(FAILWAITTIME) VALUE(0 SECONDS)”
    “MODIFY PARM NAME(WARNINGCPUTIME) VALUE(0 SECONDS)”
    “MODIFY PARM NAME(WARNINGWAITTIME) VALUE(0 SECONDS)”
```

Parameter	Description	Valid values
CHECKLIMITSINTERVAL	Specifies how often, in seconds, each client task is checked for a violation of the execution limit. The interval value is specified in seconds and should be a factor of one hour. The interval value should divide evenly into 3600 (one hour).	1 – 3600 seconds 15 seconds (default)
ERRORCPUTIME	When set to 0 seconds, the parameter is disabled.	0 seconds (default)
ERRORWAITTIME	When set to 0 seconds, the parameter is disabled.	0 seconds (default)

Parameter	Description	Valid values
FAILCPUTIME	When set to 0 seconds, the parameter is disabled.	0 seconds (default)
FAILWAITTIME	When set to 0 seconds, the parameter is disabled.	0 seconds (default)
WARNINGCPUTIME	When set to 0 seconds, the parameter is disabled.	0 seconds (default)
WARNINGWAITTIME	The external wait time limit specifies how long that a connection can remain disabled. When set to 0 seconds, the parameter is disabled.	0 seconds (default)

Enabling the program execution duration time limit mechanism

When an RPC program begins execution, the starting time is recorded. Periodically, the elapsed time for all tasks that are running RPC programs is calculated and compared to the RPC duration limit value. If an elapsed time exceeds the limit, the task in which the RPC program is running may be forced to terminate.

About this task

An exception event is scheduled before termination. An SEF EXC rule, which is scheduled to handle the event, might elect to extend the time limit and continue execution, or allow the task to be terminated abnormally.

Note: This limit is applied to total elapsed time while an RPC program is run. The program might be running normally, or it might be stalled. This limit does not test whether the RPC program is, or has, consumed CPU cycles during the elapsed time interval. The limit is applied to customer-written RPC programs. The limit is not applied to built-in CALL RPC programs available in the server or to native DB2 stored procedures that are governed by the PER-SQL time limit.

Procedure

To enable the time limit, use the `MODIFY PARM` command to add the following parameters to the `AVZSIN00` configuration member:

```
if 1 = 1 then
  do
    “MODIFY PARM NAME(RPCDURATIONLIMIT) VALUE(0 SECONDS)”
```

Parameter	Description	Valid values
RPCDURATIONLIMIT	If set to a non-zero value, the parameter imposes an elapsed time limit for all RPC program executions. The value is expressed in seconds. When set to 0 seconds, no elapsed time limitation is enforced. The maximum allowed value is 86,400 seconds, equal to 24 hours.	0 (default)

Detecting when sessions fail

The Data Virtualization Manager server can detect session failures while processing is active.

About this task

This type of failure occurs when a user submits a long-running SWL statement or RPC and then stops the application or restarts the server that hosts the application. When either condition occurs, Data Virtualization Manager server detects that the session failed and kills the SQL statement or RPC.

Procedure

To enable session failure detection, use the `MODIFY PARM` command to add the following parameters to the `AVZSIN00` configuration member:

```
if 1 = 1 then
  do
    “MODIFY PARM NAME(CHECKSESSIONS) VALUE(YES)”
    “MODIFY PARM NAME(SESSIONFAILTIME) VALUE(15 SECONDS)”
```

Parameter	Description	Valid values
CHECKSESSIONS	Controls whether to periodically check the status of each session. When a session failure is detected, all work that is running on the host on behalf of the client is terminated.	**YES** Check the status of each session periodically. **NO** (default) Do not check check the status of each session.
SESSIONFAILTIME	Specifies the number of seconds that a remote application task (a task that is running on behalf of a client) can be in processing state (RPC, SQL, REXX) before the product checks whether the network session is still active.	15 (default)

Modifying the client auxiliary storage cut-off parameter

You can specify at what point the Data Virtualization Manager server will reject new connection attempts when an auxiliary storage shortage is signaled by the system Event Notification Facility.

About this task

The Data Virtualization Manager server listens for ENF 55 auxiliary storage shortage signals and throttles storage utilization when an auxiliary storage shortage is signaled.

The Accelerator Loader server will perform the following actions depending on the received ENF 55 signal:

- When signal ENF55QLF_AUX_WARNING is received:
  1. Issue the following message:

     ```
     AVZ4265W Data Server Client buffer expansion disabled due to auxiliary storage
     warning
     ```

  2. Disable Data Virtualization Manager server buffer expansion for two hours and ten minutes.
  3. Issue the following message:

     ```
     AVZ4266I Data Server Client services resumed
     ```

- When signal ENF55QLF_AUX_SHORTAGE is received:

1. Disable Data Virtualization Manager server buffer expansion.
2. Issue the following message:

   ```
   AVZ4265W Data Server Client buffer expansion disabled due to auxiliary storage
   shortage
   ```

- When signal ENF55QLF_AUX_CRITICAL_SHORTAGE is received:
  1. Disable Data Virtualization Manager server buffer expansion.
  2. Issue the following message:

     ```
     AVZ4265W Data Server Client buffer expansion disabled due to auxiliary storage
     critical shortage
     ```

  3. Disable new Data Virtualization Manager server requests.
  4. Issue the following message:

     ```
     AVZ4267W Data Server Client refusing new requests due to critical auxiliary
     storage shortage.
     ```

- When signal ENF55QLF_AUX_SHORTAGE_RELIEVED is received:
  - Re-enable all Data Virtualization Manager server functions.
  - Issue the following message:

    ```
    AVZ4266I Data Server Client services resumed.
    ```

The point at which the Data Virtualization Manager server will reject new connection attempts when an auxiliary storage shortage is signaled by the system Event Notification Facility is controlled by the **DSCLIENTAUXSTGCUTOFF** parameter.

To change the value, complete the following steps.

Procedure

1. Locate the Data Virtualization Manager configuration member. The server initialization member is shipped in data set member *hlq*.SAVZEXEC(AVZSIN00) and may have been copied to a new data set for customization in the step "Copying target libraries" in the *Installation and Customization Guide*.
2. Use the **MODIFY PARM** command to change the **DSCLIENTAUXSTGCUTOFF** parameter value:

   ```
   "MODIFY PARM NAME(DSCLIENTAUXSTGCUTOFF) VALUE(WARNING)"
   ```

Parameter name	Parameter description	Default value
DSCLIENTAUXSTGCUTOFF	DSCLIENT AUX STORAGE NEW CONNECTION CUTOFF Specifies at what point the Data Virtualization Manager server will reject new connection attempts when an auxiliary storage shortage is signaled by the system Event Notification Facility. **WARNING** New Data Virtualization Manager server connections will be rejected when an auxiliary storage warning is received. This signal is issued when message IRA205I occurs. **SHORTAGE** New Data Virtualization Manager server connections will be rejected when an auxiliary storage shortage is signaled. This signal is issued when message IRA200E occurs. **CRITICAL** New Data Virtualization Manager server connections will not be rejected until an auxiliary storage critical shortage is signaled. This signal is issued when message IRA201E occurs.	WARNING

Running multiple servers

There are times when you might want to bring up separate Data Virtualization Manager servers, either for testing purposes or for distributing your workload.

Configuring additional Data Virtualization Manager servers

About this task

You must complete the following steps for each additional Data Virtualization Manager server you start. Sample JCL is provided in member AVZGNSUB located in the *hlq*.SAVZCNTL data set to assist with steps 1 - 5.

Procedure

1. Create a VSAM data set, for the additional Data Virtualization Manager server.
2. Make a copy of the SEF data sets for the new Data Virtualization Manager server. Copy the following data sets:

```
HLQ.ATH.SAVZEXEC
HLQ.CMD.SAVZEXEC
HLQ.EXC.SAVZEXEC
HLQ.GLV.SAVZEXEC
HLQ.PUB.SAVZEXEC
HLQ.RPC.SAVZEXEC
HLQ.SQL.SAVZEXEC
HLQ.TOD.SAVZEXEC
```

3. Optional: Use the COPYMAP step that is located in the *hlq*.SAVZCNTL(AVZGNSUB) member.

   Map data sets can be shared across subsystems. However it is advisable to have a separate map data set for each subsystem when separating application environments. If maps are shared across subsystems, you should make manual refreshes across the other subsystems when map changes are made. Map changes are also recognized at Data Virtualization Manager server installation.
4. Optional: Create and copy a new HTXLIB for Streams by using the COPYHTX step that is located in the *hlq*.SAVZCNTL(AVZGNSUB) member.

   If you run custom RPC programs, RPC data sets are required. RPCLIB.PRELOAD is an optional RPC library that you can preload RPCs into a cache.
5. Create a new startup JCL procedure. The AVZS member of the *hlq*.SAVZCNTL library contains sample JCL procedures for running the Data Virtualization Manager server main address space (started task). You should place the AVZS PROC in a procedure library where the z/OS START command searches (this may be SYS1.PROCLIB). Optionally, you can change the name of the procedure to reflect the new Server you are starting. You must change the **SSID** parameter in the startup procedure to reflect the additional subsystem name.
6. Define the new started task to the security product.
7. Obtain or define a new TCP/IP port or VTAM application ID, depending on the communication protocol you are using.
8. Create a new AVZSIN00 member. Customize this Data Virtualization Manager configuration member to configure your interfaces and run time options.
9. Create a AVZSINEF member in the SYSEXEC concatenation where AVZS refers to the new subsystem name you are creating.

   You can make a copy of your existing AVZSINEF. This member is used for initializing global variables at Data Virtualization Manager server startup. If you modified the default AVZSINEF, you should review these changes to determine whether you want them in your new Data Virtualization Manager server. If this is not done, the REXX interpreter (SDBI or SDBX command processor) is not able to locate the specified REXX program in the data set allocated to the SYSEXEC DDNAME (for SDBI) or in the specified data set. You receive a warning message at startup of the Data Virtualization Manager server, however, you can ignore this message.
10. If the two Data Virtualization Manager servers are running at different versions set up a new REXX/EXEC to invoke the ISPF application.

    Modify the LLIB statement in the REXX/EXEC to point to the Data Virtualization Manager server load library. This is not necessary if you are running at the same version but a different maintenance level. You may receive a warning regarding load library maintenance mismatches when entering the ISPF panels, however this message is informational only.

Using multiple Data Virtualization Manager servers as peers

Use the Integrated DRDA Facility (IDF) to communicate between peer Data Virtualization Manager servers.

IDF provides DRDA Application Server (AS) capability in Data Virtualization Manager, allowing communication between peer Data Virtualization Manager servers. Each Data Virtualization Manager server can use DRDA to access data sources resident at another peer Data Virtualization Manager server.

In conjunction with the DRDA Application Requestor (AR) component of Data Virtualization Manager, each Data Virtualization Manager server can operate as both a data provider and a data source.

Use IDF for peer-to-peer communications between Data Virtualization Manager servers if the servers are installed on z/OS LPARs that do not share DASD or are installed across remote z/OS locations. Using IDF, every Data Virtualization Manager server can be configured as a data source and its virtual tables made accessible to other Data Virtualization Manager servers.

To use IDF for peer-to-peer communication between your Data Virtualization Manager servers, perform the following steps:

1. Configure your Data Virtualization Manager servers. See "Configuring multiple Data Virtualization Manager servers as peers" on page 295.
2. Use the Data Virtualization Manager studio to create virtual table data maps using information provided by the remote target peer server. See "Creating virtual maps for tables on a peer server" on page 298.
3. Generate and execute requests referencing the virtual table. You can issue a query from the local server to fetch data from the remote target peer server and can include the peer table in a complex join with other local or remote tables.

Configuring multiple Data Virtualization Manager servers as peers
Configure your Data Virtualization Manager servers to use the Integrated DRDA Facility (IDF) for peer-to-peer communication.

About this task

Use the following procedure to configure multiple Data Virtualization Manager servers as peers using IDF.

The procedure uses the following terminology:

- *Local server*. The Data Virtualization Manager server that is the requester in the peer-to-peer relationship.
- *Remote target peer server*. The Data Virtualization Manager server that is the target in the peer-to-peer relationship.

DRDA is used to request and receive data from peer targets. As indicated in the procedure, you will use DEFINE DATABASE configuration commands to describe TYPE(PEER) IDF target servers.

Procedure

1. Configure the remote target Data Virtualization Manager server to activate the Integrated DRDA Facility (IDF) by setting the following start-up parameters:

```
MODIFY PARM NAME(IDF)                VALUE(YES|NO)
MODIFY PARM NAME(IDFALREADYVERIFIED) VALUE(CLIENT|SERVER)
MODIFY PARM NAME(IDFLOCATION)        VALUE(location)
MODIFY PARM NAME(IDFPORT)            VALUE(port)
MODIFY PARM NAME(OEPORTNUMBER)       VALUE(port)
```

The following table describes these server parameters:

Parameter	Description	Valid values
IDF	IDF (Integrated DRDA Facility Activated) Determines whether IDF will be activated for the current start-up or remain inactive.	**YES** Activate IDF. **NO** IDF is not active. Default: NO
IDFPORT	IDF TCP/IP MAIN PORT TCP/IP port number on which the server listens for in-bound DRDA session requests.	Any available port number appropriate for your site. Default: 50000

Parameter	Description	Valid values
IDFSSLPORT	IDF TCP/IP SSL PORT TCP/IP port number on which the server listens for in-bound DRDA SSL session requests.	Port number for SSL listener. When not set, SSL requests cannot be serviced. Default: None
IDFLOCATION	IDF LOCATION NAME DRDA location name. It is recommended that the same standard used to assign DRDA location names to Db2 subsystems be used for IDF.	A valid value is a string 1 - 16 characters. Default: '*lparssid*', where *lpar* is the SMFID of the LPAR and *ssid* is the server subsystem ID.
IDFALREADYVERIFIED	IDF ALREADY-VERIFIED SECURITY REQUIRED Specifies the minimum authentication level that can be used when a client connects to the IDF DRDA Application Server.	**SERVER** Requires a valid z/OS user ID and password. **CLIENT** Only a valid user ID must be supplied; a password is not required. Default: SERVER
OEPORT	Port number. This parameter is required for the Data Virtualization Manager studio to obtain metadata directly from the target.	

Note: It is recommended that the other IDF-related parameters be allowed to default unless you have specific requirements for their use.

2. On the local server, define each remote target peer server as a peer-type database server, as follows:

```
DEFINE DATABASE TYPE(PEER)
       NAME(name)
       DDFSTATUS(ENABLE)
       DOMAIN(your.domain.name)
       SECMEC(USRIDONL)
       LOCATION(location)
       PORT(port)
       OEPORT(port)
       CCSID(ccsid)
```

The following table describes these parameters:

Parameter	Description	Valid values
CCSID	The EBCDIC single-byte application CCSID (Coded Character Set Identifier). This value must match the `SQLENGDFLTCCSID` value specified for the target server. (*Required*)	Refer to the data source vendor documentation for a list of valid CCSID values.

Parameter	Description	Valid values
DDFSTATUS	The DDF activation status. (*Required*)	**ENABLE** Make this DDF definition active. **DISABLE** DDF endpoint is not used.
DOMAIN	The domain name or hostname on which the Data Virtualization Manager server is running. Either DOMAIN or IPADDR is required, but not both.	No default value
IPADDR	The dot-notation IPV4 or IPV6 address of the host on which the Data Virtualization Manager server is running. Either DOMAIN or IPADDR is required, but not both.	
LOCATION	DRDA location name. This value must match the `IDFLOCATION` value specified for the target server.	A valid value is a string 1 - 16 characters.
NAME	Target server name.	A valid value consists of 1 - 4 characters.
PORT	This value must match the `IDFPORT` value specified for the target server.	1 - 65535
OEPORT	This value must match the `OEPORT` value specified for the target server.	1 - 65535
SECMEC	The DRDA security mechanism in force. This value depends on the `IDFALREADYVERIFIED` value specified for the target server.	**USRIDPWD** User ID and password are sent as is. No encryption is used. Use this setting if `IDFALREADYVERIFIED` is set to SERVER for the target server. **USRIDONL** User ID is sent as is. No encryption is used for the user ID only (client security). Use this setting if `IDFALREADYVERIFIED` is set to `CLIENT` for the target server.
TYPE	Defines the DDF endpoint type. **PEER** DDF endpoint is an IDF target server.	When using IDF for peer-to-peer communication between servers, PEER is the valid value.

Example

The following example shows the settings in a peer-to-peer server configuration using IDF.

On the remote target peer server, the following start-up parameters are defined:

```
MODIFY PARM NAME(IDF)                  VALUE(YES)
MODIFY PARM NAME(IDFALREADYVERIFIED) VALUE(CLIENT)
MODIFY PARM NAME(IDFLOCATION)        VALUE(ZOS1RDBF)
MODIFY PARM NAME(IDFPORT)            VALUE(9999)
MODIFY PARM NAME(OEPORT)             VALUE(9991)
MODIFY PARM NAME(SQLENGDFLTCCSID)    VALUE(1047)
```

On the local server, the remote target peer server is defined as follows:

```
DEFINE DATABASE TYPE(PEER)
       NAME(RDBF)
       DDFSTATUS(ENABLE)
       DOMAIN(zos1.domain.name)
       SECMEC(USRIDONL)
       LOCATION(ZOS1RDBF)
       PORT(9999)
       OEPORT(9991)
       CCSID(1047)
```

The following table summarizes how the values correlate:

Server start-up parameter on remote target peer server	Database definition for remote target peer server on local server
IDFALREADYVERIFIED	SECMEC
IDFLOCATION	LOCATION
IDFPORT	PORT
OEPORT	OEPORT
SQLENGDFLTCCSID	CCSID

What to do next
Use the Data Virtualization Manager studio to create virtual table data maps using information provided by the remote target peer server. See "Creating virtual maps for tables on a peer server" on page 298.

Creating virtual maps for tables on a peer server
Create virtual data maps for tables defined on a remote target peer server.

Before you begin
The Data Virtualization Manager server and target peer server must be configured as peer servers. See "Configuring multiple Data Virtualization Manager servers as peers" on page 295.

About this task

Using the Integrated DRDA Facility (IDF), every Data Virtualization Manager server can be configured as a data source and its virtual tables made accessible to other Data Virtualization Manager servers. Use the Data Virtualization Manager studio to create virtual table data maps using information provided by the remote target peer server.

Procedure

1. On the **Server** tab, expand the **Discovery** > **Peer Subsystems** > ***SSID*** node, where *SSID* is the name of your remote target peer server.
2. Select the virtual tables, views, or both, that you want to use, and then right-click and choose **Create Virtual Table(s)**.

3. In the **New Virtual Tables Wizard**, on the **New Virtual Tables for Peer Subsystem access** page, complete the following fields:

Field	Description
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, *hlq*.USER.MAP). The target libraries are specified in the server's started task JCL.
Description	Enter an optional description.
Naming Pattern	Specify the format to use for the generated virtual table names. Use the following variables to create naming patterns that are derived from the metadata: • {Subsystem}: Subsystem name • {Table}: Source table name
Virtual Target System	Select a virtual target system from the drop-down list. A virtual target system points to the subsystem that contains the data that you want to access using the current virtual table. If there are no virtual target systems in the drop-down list, click **Create Target System** to create one. By using virtual target systems, you can easily change the name of the subsystem that is referenced in the virtual tables. For example, on a local server AVZ1, you create a virtual target system called TSIDF as the IDF target system, and specify that it will access the remote subsystem peer AVZ2. Then, you create 50 virtual tables that access data in the source TSIDF (that is, pointing to AVZ2). If it becomes necessary to change the name of the source AVZ2, you only have to change it in a single place by editing the virtual target system. These target systems can be located under the **SQL** > **Target Systems** > **DBMS** node in the server view tree.

4. In the results table, review the list of selected entries. Modify the selections as needed.

   Tip: Use the check box in the header row of the table to control the selection of all entries.
5. To disable MapReduce, click **Advanced** and select **Disable MapReduce**.
6. Click **Finish**.

Results
The studio creates the virtual tables (the metadata maps) on the local server. The virtual tables appear under the **SQL** > **Data** > ***SSID*** > **Virtual Tables** tree node, where *SSID* is the name of the local server.

What to do next
Generate and execute requests referencing the virtual table. You can issue a query from the local server to fetch data from the remote target peer server and can include the peer table in a complex join with other local or remote tables.

z Systems Data Compression (zEDC)
IBM z Systems Data Compression (zEDC) is an accelerated compression solution that provides high performance, low latency compression with minimal system overhead.

zEDC uses an industry standard compression library that provides efficient performance with large sequential files. zEDC facilitates cross-platform exchange of data.

Enabling zEDC
Data Virtualization Manager server provides support for IBM z Systems Data Compression (zEDC).

Before you begin
To determine the hardware and software requirements, refer to the current *IBM z Systems Data Compression* documentation.

Procedure

1. Set `NETWORKBUFFERSIZE` on both Data Virtualization Manager servers to a value between `ZEDCMINDATASIZE` and `1048512`.
2. Set the following parameters in the `AVZSIN00` configuration member:

```
/*----------------------------------------------------------------------*/
/* Enable ZEDC support.                                                 */
/*----------------------------------------------------------------------*/
if 1 = 1 then
   do
     "MODIFY PARM NAME(ZEDCCOMPRESSION) VALUE(YES)"
     "MODIFY PARM NAME(ZEDCMINDATASIZE) VALUE(8192)"
end

if 1 = 1 then
   do
     "MODIFY PARM NAME(TRACEZEDCCOMPRESSION) VALUE(NO)"
     "MODIFY PARM NAME(TRACEFULLZEDC) VALUE(NO)"
end
```

The following table lists the parameters for enabling zEDC:

Parameter	Description	Valid values
NETWORKBUFFERSIZE	Controls the size of the buffer used to receive blocks of data from the network. A failure will occur if a client application sends a buffer larger than the maximum size. This value should be raised to allow larger blocks of data to be sent to and from the client.	256K (default) or required size.
TRACEZEDCCOMPRESSION	Enables tracing of all zEDC calls to the Server Trace facility. It should only be set to YES if the user needs to trace zEDC calls for diagnostic purposes.	**YES** Enable zEDC tracing. **NO** (default) Do not enable zEDC tracing.
TRACEFULLZEDC	Traces the entire buffer, not just the first few bytes. It should only be set to YES if a minimal trace is not enough.	**YES** Enable zEDC tracing for the entire buffer. **NO** (default) Do not enable full zEDC tracing.
ZEDCCOMPRESSION	Enables or disables the use of the zEDC compression hardware device. Set to YES if you have the zEDC compression hardware and wish to use it.	**YES** Enable zEDC compression. **NO** (default) Do not enable zEDC compression.
ZEDCMINDATASIZE	Sets the minimum amount of data the server will compress with the zEDC hardware. It is recommended that testing first be done with a minimum size of 8K.	8192 (default) or required size.

3. To verify that zEDC is in use, enable zEDC tracing (`TRACEZEDCCOMPRESSION`) and look for ZED events in the Server Trace.

IDF for DB2 Mainframe Applications

You can use the Integrated DRDA Facility (IDF) for DB2 mainframe applications so that z/OS DB2 systems can communicate with Data Virtualization Manager server to access Z and distributed data sources using Distributed Relational Database Architecture (DRDA).

Support of this feature is based upon the use of three-part table names within DB2 stored procedures and applications, DB2 RESTful Services, QMF for TSO, and the SQL processor using file input (SPUFI) facility within the DB2 Interactive (DB2I) primary option menu of ISPF. Table names in SQL statements are composed of three parts: the location name, the schema or owner name, and the table name; the parts being delimited by periods. For example, *LOCATION.SCHEMA.TABLE* is a fully-qualified three-part table name. Often, the first, or first two parts of a table name are omitted with default values assumed for each.

When the first part of a three-part table name is omitted, the current DB2 Server is the assumed location where the table resides. Using fully qualified three-part table names, the location name part may designate the same or some other server where the table resides. When a table is remote, DRDA is used to communicate between DB2 and the remote server. The SQL is sent to the remote server for execution and the results returned to the requestor.

Note: DB2 supports definition of a table alias that maps to a three-part table name, thereby avoiding explicit, hard-coded location designations within applications and stored procedures. These aliases can later readily be re-assigned to change from one location to another or from a test to a production environment.

IDF allows a Data Virtualization Manager server to participate as an endpoint in multi-homed configurations. Whenever SQL is executed in DB2 and a three-part table name designates an IDF endpoint, the SQL is sent to the Data Virtualization server for execution. There, the table name designates a virtual table which may be backed by VSAM, ADABAS, IMS, or any of the other supported backend sources Data Virtualization Manager supports. Results of the execution are returned to DB2, which proceeds as it would for any local table access.

Known Restrictions and Limitations

IDF for DB2 mainframe applications supports SELECT (including SELECT INTO), INSERT, UPDATE, and DELETE access to data sources using either bound-statements or dynamic SQL.

Note: The operations supported vary by data source. Not all data sources allow for all update-type operations. For example, you cannot alter data within a logstream in any way, nor you cannot delete rows from a sequential dataset.

Stored procedures and DB2 application programs may execute statements bound into a package or SQL statements can be prepared dynamically. DB2 applications can use the DCLGEN utility to extract meta-data information from the Data Virtualization Manager server. DB2 Bind is used to propagate packages and SQL statements to target Data Virtualization Manager servers. The facility does not yet support scrollable cursors, cursor-positioned updates, nor two-phase commit.

Following are the known restrictions and limitations of IDF:

- **Restrictions:**
  - IDF supports INSERT, DELETE, UPDATE, SELECT, and SELECT INTO statements. For pathways using a bound package, the facility supports the use of DECLARE CURSOR FOR SELECT statements, along with OPEN, FETCH, and CLOSE for the cursor.
  - The operations allowed may differ for each data source. For example, Data Virtualization Manager server virtual tables backed by logstreams cannot be updated.
  - COMMIT and ROLLBACK operations may not work as expected due to the limitations of the back-end virtual table data source. For example, ROLLBACK may not work as expected when using VSAM

datasets unless the RLS recovery attributes of the dataset meet the requirements needed for backout processing.
  - You should ensure that the IDF SQL statements conform to ANSI-92 standards. That is, you should specify SQL92 for language level in the DB2 preprocessor. The SQL engine only supports apostrophes and period. You may achieve this by using the pre-compiler parm SQLFLAG(SQL92E).
- **Limitations:**
  - SQLCODE and SQLSTATE values returned by execution of SQL statements over IDF may differ from codes returned by DB2. In many cases, the facility will return the non-specific SQLCODE -1 with more specific error text returned in the SQLERRMC fields.
  - When DB2 applications run, and if DB2 requests IDF not to update any resources, IDF deliberately ignores the do-not-update request and instead allows the updates to occur and engages auto-commit processing after each operation.
  - Versioning for DB2 stored procedures and for DB2 RESTful service definitions is not supported.
  - Cursor positioned UPDATE and DELETE, that is “WHERE CURRENT OF CURSOR” form, are not supported.
  - Two-phase commit is not supported.

Configuring Data Virtualization Manager server for IDF

Data Virtualization Manager server requires some DRDA Application Server setup parameters for operation. Perform the following step to configure a Data Virtualization Manager server for IDF:

Procedure

To activate the DRDA-AS (Application Server) to handle IDF requests in a Data Virtualization Manager server, set the following startup parameters in the IN00 dataset:

Parameter Name	Description	Default value	Value to Set
IDF	Determines whether IDF/IDF will be activated for the current start-up or remain inactive.	**NO** IDF is not active.	***Must*** be set to YES to activate IDF
IDFPORT	TCP/IP port number on which the server listens for in-bound DRDA session requests.	50000	Choose any available port number appropriate for your site.
IDFSSLPORT	TCP/IP port number on which the server listens for in-bound DRDA SSL session requests.	None	Choose a port number for SSL listener. When unset, SSL requests cannot be serviced.
IDFLOCATION	DRDA location name.	The LPAR’s SMFID suffixed with server’s subsystem ID to create an 8-byte location name	We recommend that same standard used to assign DRDA location names to DB2 subsystems be used for IDF.

Parameter Name	Description	Default value	Value to Set
IDFALREADYVERIFIED	Specifies the minimum authentication level that can be used when any DRDA client connects to the IDF/VDF DRDA Application Server.	SERVER	SERVER - require a valid z/OS User ID and password. CLIENT - only a valid User ID must be supplied; a password is not required.
PASSTICKETAPPNAME	Specifies the APPLNAME which the server uses to validate RACF passtickets.	The server's three-byte product ID suffixed with the LPAR's SMFID	If RACF passtickets are to be sent by DB2 when connecting to IDF, this value **MUST** match the LINKNAME value used to populate the corresponding DB2 CDB communications database tables.
It is recommended that the other IDF-related parameters be set with their default values unless you have specific requirements for their use.			

Populating DB2 communications database for IDF

Before DB2 can connect to a remote IBM Data Virtualization Manager for z/OS server, DB2 must know that a location name (from the three-part table name in SQL) is valid and it must know how to establish a TCP/IP communication session between itself and the peer location. DB2 refers to a set of catalog tables, collectively known as communication database (CDB), to resolve location names to the information needed to establish DRDA sessions.

It is recommended that you review the Populating the communications database for use with TCP/IP topic for more information on how to populate CDB.

Each remote location is defined to DB2 by inserting or updating rows in the following catalog tables:

- SYSIBM.LOCATIONS
- SYSIBM.IPNAMES
- SYSIBM.USERNAMES

Populating the SYSIBM.LOCATIONS table
Each IBM Data Virtualization Manager for z/OS server must have a unique IDFLOCATION name assigned to it. The assigned location name is the high-order qualifier used in three-part table names. At minimum, one row must be added to SYSIBM.LOCATIONS to define the location to DB2. The table LOCATION column is a primary key and must match the unique IBM Data Virtualization Manager for z/OS server target IDFLOCATION. The row's LINKNAME column is used to de-reference connection information in the other tables. A LINKNAME value may be uniquely defined for the specific location's connection information, or it may designate a set of definitions used in common for accessing many remote servers. Perform the following to populate the SYSIBM.LOCATIONS table:

Procedure

Set the columns of the SYSIBM.LOCATIONS catalog table to:

Column	Value
LOCATION	Set to the unique IDFLOCATION name assigned to the target DV server

Column	Value
LINKNAME	An 8-byte link name value. This name is used to look up matching entries in the other SYSIBM tables.
PORT	Set to the value of IDFPORT or IDFSSLPORT.
SECURE	Set to “Y” if SSL is to be used (PORT should be set to IDFSSLPORT number)

Following is a sample SQL statement used to create a row in the locations table:

```
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, PORT, SECURE)
       VALUES(‘LQ52ABCD’, ‘LQ52ABCD’, ‘7000’, ‘N’)
```

Populating the SYSIBM.IPNAMES table
A row in the SYSIBM.IPNAMES table contains to set up information for the remote server’s IP address and security processing used when connecting. The row in SYSIBM.IPNAMES is selected by matching to the LINKNAME from the SYSIBM.LOCATIONS row for the target server. Perform the following to populate the SYSIBM.IPNAMES table:

Procedure

Set the columns of the SYSIBM.IPNAMES catalog table to:

Column	Value
LINKNAME	An 8-byte link name value. This name is used to matched to the LINKNAME in the SYSIBM.LOCATIONS table.
SECURITY_OUT	IDF for DB2 applications supports the use of options ‘A’, ‘P’, and ‘R’. For more information on these options, For more information on IPNAMES catalog table, refer the IPNAMES topic. **Note:** IDF does not support DRDA-level encryption of data *payload* objects (options D and E are not supported). SSL sessions should be used when data *payload* encryption is needed. IDF support the use of encrypted credentials. When you select option A, P, or R, DB2 will initially request to use clear-text credentials. IDF can be configured to require that user ID, or user ID and password be AES encrypted, and will negotiate this added level of protection with DB2 before the values are transmitted.
USERNAMES	Set ‘O’ to use SYSIBM.USERNAMES translation or blank to pass user IDs unchanged
IPADDR	Set to the IPv4 or IPv6 IP address or the character format domain name of the remote location.

Following is a sample SQL statement used to create a row in the IP names table:

```
INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IPADDR)
       VALUES(‘LQ52ABCD’, ‘A’, ‘O’, ‘LQ52’)
```

Populating the SYSIBM.USERNAMES table

A row in the SYSIBM.USERNAMES table contains to set up information used to translate/output user IDs send when connecting to the remote server. The row in SYSIBM.USERNAMES is selected by matching to the AUTHID and LINKNAME columns. Perform the following to populate the SYSIBM.USERNAMES table:

Procedure

Set the columns of the SYSIBM.USERNAMES catalog table to:

Column	Value
LINKNAME	An eight-byte link name value. This name is used to matched to the LINKNAME in the SYSIBM.LOCATIONS table.
TYPE	Specify ‘O’ for outbound security translation
AUTHID	Set to a specific user ID or to blanks to match to all user IDs.
NEWAUTHID	New authorization ID to be output in place of AUTHID from lookup.
PASSWORD	Password to accompany translated authorization ID.

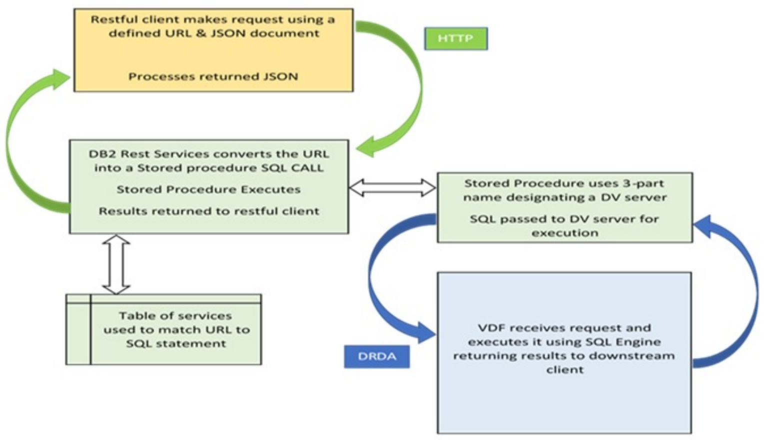
Following is a sample SQL statement used to create a row in the IP names table:

```
INSERT INTO SYSIBM.USERNAMES (LINKNAME, TYPE, AUTHID)
       VALUES(‘LQ52ABCD’, ‘O’, ‘APJKK’)
```

Using IDF for Mainframe Applications to implement DB2 RESTful services

When creating DB2 RESTful services, IDF can be used to incorporate data virtualization virtualized tables and views. Using Data Virtualization Manager server, you can create UDTFs which are inherently read-only. Using IDF for mainframe applications, you can create a DB2 Stored Procedure that uses three-part table names to both read and update non-DB2 data sources. This stored procedure can then be incorporated into a DB2 REST Service

The following diagram illustrates the access flow when IDF three-part table name is used in a DB2 stored procedure to implement a DB2 RESTful service.


Restful client makes request using a defined URL & JSON document
Processes returned JSON
HTTP
DB2 Rest Services converts the URL into a Stored procedure SQL CALL
Stored Procedure Executes
Results returned to restful client
Stored Procedure uses 3-part name designating a DV server
SQL passed to DV server for execution
Table of services used to match URL to SQL statement
DRDA
VDF receives request and executes it using SQL Engine returning results to downstream client

DB2 RESTful services provisioning

One use case of IDF for DB2 mainframe applications is in the creation of RESTful web services using three-part table names within DB2 stored procedures. If this is an intended use case, refer DB2 REST services for information about provisioning DB2 REST services.

Note: IDF for mainframe applications does not support versioning for DB2 REST services, nor versioning for DB2 stored procedure packages.

Creating a DB2 RESTful service using IDF

When creating a DB2 REST service, you cannot use three-part table names directly in the SQL text bound with the DB2 REST service definitions. Three-part table name references are not supported for use in the SQL text for these services. However, this restriction applies only to the *bound-within-the-service* SQL statement text. A DB2 REST service may invoke a local stored procedure using an SQL CALL statement, and that local stored procedure *may* contain embedded three-part table name references.

About this task

The steps to define a DB2 REST service using IDF for mainframe applications are:

1. Create a DB2 stored procedure using three-part table name references to Data Virtualization Manager server-based tables.
2. Bind or copy the stored procedure package into the Data Virtualization Manager server.
3. Define the RESTful service as an SQL CALL invoking the procedure.

Example

The following sample JCL illustrates one means of creating a DB2 stored procedure using batch JCL. You can also use the SQL processor using file input (SPUFI) facility within the DB2 Interactive (DB2I) primary option menu of ISPF or run the CREATE procedure from a remote studio application. The procedure parameters are used to insert a row into a Data Virtualization Manager server managed VSAM dataset.

```
//JOBCARD  JOB ,SAMPLE,CLASS=A,MSGCLASS=X,REGION=4M,
//  NOTIFY=&SYSUID
//*  Create stored procedure that inserts row into IDF table at LQ52ABCD server
//BIND1      EXEC  PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB    DD    DISP=SHR,DSN=DB9D.SDSNEXIT
//           DD    DISP=SHR,DSN=DSN.VB10.SDSNLOAD
//           DD    DISP=SHR,DSN=DSN.VB10.RUNLIB.LOAD
//DBRMLIB    DD    DISP=SHR,DSN=DSN.VB10.DBRMLIB.DATA
//SYSPRINT   DD    SYSOUT=*
//SYSTSPRT   DD    SYSOUT=*
//SYSUDUMP   DD    SYSOUT=*
//SYSTSIN    DD    *
  DSN SYSTEM(DB9D)
  RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP2)
  END
/*
//SYSIN    DD *
--#SET TERMINATOR &
  DROP PROCEDURE SYSPROC.PUTORG&
  COMMIT&
  CREATE PROCEDURE SYSPROC.PUTORG(IN  PDEPT  INT,
                                  IN  PMGR   INT,
                                  IN  PNAME  CHAR(20),
                                  IN  PDIV   CHAR(10),
                                  IN  PLOC   CHAR(20))
      LANGUAGE SQL
  BEGIN
      INSERT INTO LQ52ABCD.VDFDEMO.ORGSEQ
             (DEPTNUMB, MANAGER, DEPTNAME, DIVISION, LOCATION)
             VALUES(PDEPT, PMGR, PNAME, PDIV, PLOC);
      COMMIT;
  END&
  GRANT EXECUTE ON PROCEDURE SYSPROC.PUTORG TO PUBLIC&
  COMMIT&
/*
```

When a stored procedure is created, DB2 internally generates a package containing the SQL statements from within the procedure. The stored procedure package must be copied from DB2 into the target IDF Data Virtualization Manager server. You can use SPUFI bind-package dialogs to perform the bind-copy operation, or utilize a batch job as illustrated in the following sample:

```
//JOBCARD  JOB ,SAMPLE,CLASS=A,MSGCLASS=X,REGION=4M,
//  NOTIFY=&SYSUID
//*  Copy SP package to IDF server LQ52ABCD
//BIND2      EXEC  PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB    DD    DISP=SHR,DSN=DB9D.SDSNEXIT
//           DD    DISP=SHR,DSN=DSN.VB10.SDSNLOAD
//           DD    DISP=SHR,DSN=DSN.VB10.RUNLIB.LOAD
//SYSPRINT   DD    SYSOUT=*
//SYSTSPRT   DD    SYSOUT=*
//SYSUDUMP   DD    SYSOUT=*
//SYSTSIN    DD    *
  DSN SYSTEM(DB9D)
  BIND                          -
   ACTION(REPLACE)              -
   CURRENTDATA(NO)              -
   COPY(SYSPROC.PUTORG)         -
   COPYVER(V1)                  -
   PACKAGE(LQ52ABCD.SYSPROC)    -
   DEFER(PREPARE)               -
   KEEPDYNAMIC(YES)             -
   EXPLAIN(NO)                  -
   FLAG(I)                      -
   DEGREE(1)                    -
   ISOLATION(CS)                -
   RELEASE(COMMIT)              -
   SQLERROR(CONTINUE)           -
   REOPT(ONCE)                  -
   VALIDATE(BIND)               -
   DYNAMICRULES(RUN)            -
   ENCODING(EBCDIC)             -
   APREUSE(NONE)                -
   APCOMPARE(NONE)
  END
/*
```

Note:

- The COPYVER version must be V1.
- IDF does not support stored procedure package versioning.

Finally, you must define the RESTful service URL and relate it to an SQL statement that is executed when the service is requested. The following sample shows this definition being performed using batch JCL. You may also use a DB2-provided RESTful service to create the definition.

```
//JOBCARD  JOB ,SAMPLE,CLASS=A,MSGCLASS=X,REGION=4M,
//  NOTIFY=&SYSUID
//*  Define the REST service to DB2
//BIND3      EXEC  PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB    DD    DISP=SHR,DSN=DB9D.SDSNEXIT
//           DD    DISP=SHR,DSN=DSN.VB10.SDSNLOAD
//           DD    DISP=SHR,DSN=DSN.VB10.RUNLIB.LOAD
//SQLDDNAM DD *
    call sysproc.Putorg(:pdept, :pmgr, :pname, :pdiv, :ploc)
//SYSPRINT   DD    SYSOUT=*
//SYSTSPRT   DD    SYSOUT=*
//SYSUDUMP   DD    SYSOUT=*
//SYSTSIN    DD    *
  DSN SYSTEM(DB9D)
   BIND SERVICE(VDFDEMO) NAME("Putorg") -
   SQLDDNAME(SQLDDNAM) -
   DESCRIPTION('Add a row to ORGSEQ table')
  END
/*
```

Note that the SQL statement is a CALL for the storage procedure with the parameters defined when the stored procedure was created.

Once this definition is made, you can execute the RESTful service by posting a JavaScript Object Notation (JSON) document containing values for each of the parameters to the `/service/VDFDEMO/Putorg`URL.

Accessing IDF from DB2I (SPUFI)

Using the SQL processor using file input (SPUFI) facility to connect to DB2, you can issue SQL queries that uses 3-part names to access IBM Data Virtualization Manager for z/OS via Integrated DRDA Facility (IDF).

Example

The following selects 5 rows from a VSAM dataset mapped as STAFFVS by the IDF server:

```
  SELECT * FROM LQ52ABCD.VDFDEMO.STAFFVS FETCH FIRST 5 ROWS ONLY;
---------+---------+---------+---------+---------+---------+---------+-
    ID  NAME_L  NAME_D       DEPT  JOB       YRS
---------+---------+---------+---------+---------+---------+---------+-
    10       7  SANDERS        20  MGR         7
    20       6  PERNAL         20  SALES       8
    30       8  MARENGHI       38  MGR         5
    40       7  O'BRIEN        38  SALES       6
    50       5  HANES          15  MGR        10
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+-
---------+---------+---------+---------+---------+---------+---------+-
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+-
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 32
```

You can readily create a table alias so that the details of the three-part table name are hidden. In the following example, the alias VSTAFF is created and used instead of the fully-qualified three-part table name:

```
  CREATE ALIAS VSTAFF FOR TABLE LQ52ABCD.VDFDEMO.STAFFVS;
---------+---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+-
  COMMIT;
---------+---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+-
  SELECT * FROM VSTAFF FETCH FIRST 5 ROWS ONLY;
---------+---------+---------+---------+---------+---------+---------+-
    ID  NAME_L  NAME_D       DEPT  JOB       YRS
---------+---------+---------+---------+---------+---------+---------+-
    10       7  SANDERS        20  MGR         7
    20       6  PERNAL         20  SALES       8
    30       8  MARENGHI       38  MGR         5
    40       7  O'BRIEN        38  SALES       6
    50       5  HANES          15  MGR        10
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+-
---------+---------+---------+---------+---------+---------+---------+-
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+-
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 3
DSNE621I NUMBER OF INPUT RECORDS READ IS 35
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 61
```

You do not need to bind-copy the DB2I package into IDF before using SPUFI. IDF allows statements to be dynamically prepared in absence of a stored or bound package definition. All SQL statements entered using SPUFI are dynamically prepared.

Using IDF in DB2 application programs

About this task

You can also use the three-part table name references in DB2 applications such as COBOL batch programs. To do so, some of the following steps will be necessary depending on the requirements of the application:

1. Create table aliases for three-part table names. This may serve to make the application more portable. The aliases can be changed without re-coding the application.
2. Use the DCLGEN utility in DB2 Interactive (DB2I) to extract table column definitions for the virtual tables within the Integrated DRDA Facility (IDF) server.
3. Pre-process the application program through the DB2 pre-compiler utility.
4. Compile and link the application.
5. Bind the DataBase Request Module (DBRM) into a plan or a package in the local DB2 system.
6. Bind or copy the package to the IDF server.

Example

The following is an abbreviated COBOL application sample program. The program generates nonce data from the execution date/time and inserts a row into the virtual table hosted on the IDF server.

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.    SAMPORG.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
******************************************************************
* DCLGEN TABLE(LQ52ABCD.VDFDEMO.ORGSEQ)                          *
*        LIBRARY(CSD.AI38.W5.COBOL(ORGSEQ))                      *
*        ACTION(REPLACE)                                         *
*        LANGUAGE(COBOL)                                         *
*        STRUCTURE(ORGTAB)                                       *
*        QUOTE                                                   *
* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS   *
******************************************************************
     EXEC SQL DECLARE LQ52ABCD.VDFDEMO.ORGSEQ TABLE
     ( DEPTNUMB                       SMALLINT NOT NULL,
       MANAGER                        SMALLINT NOT NULL,
       DEPTNAME                       CHAR(20) NOT NULL,
       DIVISION                       CHAR(10) NOT NULL,
       LOCATION                       CHAR(20) NOT NULL
     ) END-EXEC.
******************************************************************
* COBOL DECLARATION FOR TABLE LQ52ABCD.VDFDEMO.ORGSEQ            *
******************************************************************
 01  ORGTAB.
     10 DEPTNUMB             PIC S9(4) USAGE COMP.
     10 MANAGER              PIC S9(4) USAGE COMP.
     10 DEPTNAME             PIC X(20).
     10 ODIVISION            PIC X(10).
     10 LOCATION             PIC X(20).
01  H1-DATE.
     05  H1-YYYY   PIC 9(4).
     05  H1-MM     PIC 9(2).
     05  H1-DD     PIC 9(2).
 01  H1-TIME.
     05  H1-HH     PIC 9(2).
     05  H1-MN     PIC 9(2).
     05  H1-SS     PIC 9(2).
     05  H1-TT     PIC 9(2).
 PROCEDURE DIVISION.
     ACCEPT H1-DATE FROM DATE YYYYMMDD.
     ACCEPT H1-TIME FROM TIME.
     COMPUTE DEPTNUMB        = H1-TT + H1-MN + H1-HH.
     COMPUTE MANAGER         = H1-YYYY + H1-MM + H1-DD.
     MOVE 'GENERATED ENTRY' TO DEPTNAME.
     MOVE H1-TIME           TO ODIVISION.
     MOVE 'PGM=SAMPORG'     TO LOCATION.
     EXEC SQL
       INSERT INTO LQ52ABCD.VDFDEMO.ORGSEQ VALUES(
       :DEPTNUMB, :MANAGER, :DEPTNAME, :ODVISION, :LOCATION)
     END-EXEC.
     GOBACK.
```

Managing user connections

Display information about user connections and terminate connections.

About this task

The Remote User panel displays current and cumulative information about users who are connected to the Data Virtualization Manager server.

Procedure

1. From the Primary Option Menu panel, enter A on the Option line.
2. On the Remote User panel, type one of the following line commands, and press Enter:

Command	Description
A	Lists the DB2 secondary authorization IDs.
C	Cancels the thread.
E	Starts the SQL EXPLAIN program, which requires MVS/Quick-Ref.
F	Formats the row.
I	Displays user information.
K	Terminates the user's connection with the Data Virtualization Manager server. To use this command, you must have UPDATE authority to the USERS resource or be using the same user ID as the connection that is being terminated. The task that supports the remote client fails with an X '222' abend. A reason code is not associated with this event.
P	Prints the control block.
S	Displays the control block.
T	Displays user trace. User trace provides summary information about user activity.
U	Displays user detail. User details provide resource utilization information.

Remote User panel

The following table describes each column name on the Remote User panel and provides a sort name (if available) for each column.

Note: Use the **PF10** key to scroll LEFT and **PF11** key to scroll RIGHT when viewing the columns in the panel.

Column name	Description	Sort name
HOST USERID	The user ID of the remote user.	USER
LAN USERID	The LAN user ID of the remote user.	LAN

Column name	Description	Sort name
HOST NAME	The link that is being used. For a local user, the name of the remote system that is being accessed. For a remote user, the name of the remote system that is accessing the local system.	HOST
LINK TYPE	The communication protocol.	TYPE
APPLICATION NAME	The application name that is specified in the APNA (Application Name) keyword of the ODBC data source.	APPLICATION
USER PARAMETER	The parameter that is specified in the USERPARM (Host User Parm) keyword of the ODBC data source.	PARAMETER
TCP/IP ADDRESS	The 4-byte IP address of a node.	IPADDR
REMOTE PORT	The port that is used by the remote Data Virtualization Manager server.	REMOTE
LOCAL PORT	The TCP/IP port that is used by the remote Data Virtualization Manager server.	LOCAL
IUCV PATH	The token that is used by Data Virtualization Manager server to communicate with TCP/IP.	PATH
SOCKET NUMBER	The number of the TCP/IP session.	SOCKET
DB2 SUBS	The DB2 subsystem to which the remote user is connected.	DB2
PLAN NAME	The name of the plan that was used to open a DB2 thread.	PLAN
SQL REASON	The most recent DB2 reason code.	REASON
SQL CODE	The most recent SQLCA SQLCODE value.	SQLCODE
SQL STMT-TYPE	The SQL verb.	SQLTYPE
STATEMENT NUMBER	The SQL statement number assigned by the preprocessor.	STMTNO
CURSOR NUMBER	The number of the cursor that is being used.	CURSOR
LOCKS HELD	The number of locks held.	
PROGRAM NAME	The Data Virtualization Manager server transaction program name.	PROGRAM
CPU TIME	The total amount of CPU time.	

Column name	Description	Sort name
SQL COUNT	The total number of SQL operations, which includes SQL run, RPCs and stored procedures that are run, rollbacks and commits initiated from the client through an ODBC call, and operations to turn auto-commit on or off.	SQLCOUNT
CONNECT TIME	The total amount of elapsed time.	CONNECT
CURRENT STATE	One of the following states: PROCESS, SEND, RECEIVE.	STATE
STATE DURATION	The amount of time that is spent in the current state.	DURATION
FUNCTION CODE	The type of Data Virtualization Manager server processing.	FUNCTION
GENERIC USERID	The DB2 generic ID for the connection.	GENERIC
DB2 G-MBR	The group attachment member name.	G-MBR
DB2 TOKEN THREAD	The DB2 token thread value.	TOKEN
DB2 REQUESTING SITE NAME	The DB2 requesting site name.	
TOTAL SENT (KB)	Cumulative outbound data. One of the following: RAW (before compression) COMPRESSED (after compression) PERCENT (1–(COMPRESSED/RAW))*100	TOSENTR TOSENTC TOSENTP
CURRENT SENT	Last outbound transmission. One of the following: RAW (before compression) COMPRESSED (after compression) PERCENT ((1–(COMPRESSED/RAW))*100)	CUSENTR CUSENTC CUSENTP
TOTAL RECEIVED (KB)	Cumulative inbound data. One of the following: RAW (before compression) COMPRESSED (after compression) PERCENT ((1–(COMPRESSED/RAW))*100)	TORECVR TORECVC TORECVP

Column name	Description	Sort name
CURRENT RECEIVED	Last inbound transmission. One of the following: RAW (before compression) COMPRESSED (after compression) PERCENT ((1–(COMPRESSED/RAW))*100)	CURECVR CURECVC CURECVP
TELEPROCESSING PERCENT	The percentage of the total data transfer time.	TPPERCNT
HOST PROCESSING PERCENT	The percentage of the total data extraction time.	HOPERCNT
ENCLAVE COUNT	The number of WLM enclaves created for the user.	COUNT
ENCLAVE CPU	The enclave CPU time, in seconds.	ENCLAVE
TOTAL CPU	The total amount of CPU time, in seconds.	CPU
ACEE SOURCE	The source of the ACEE.	ACEE
EXTENDED CLIENT	The extended client user ID, workstation name, and application name.	EXTENDED

Terminating a user connection

In the Remote User program, the **`Kill`** command is used to terminate a remote user's connection with the Data Virtualization Manager server. This command closes the TCP/IP session along with the driver. When a remote user's connection is terminated, the task supporting the remote client fails with an X '222' abend. A reason code is not associated with this event.

Before you begin

The **`Kill`** command can only be issued by a user with authorization to do so. Authorization is granted in either of the following cases:

- When the user is granted UPDATE authority to the USERS resource.
- When the user ID attempting to kill the connection is the same as the user ID of the driver being killed. In this case, UPDATE authority is not checked.

About this task

The Server Trace program shows the following information:

- Authorization request for the kill operation
- Abend of the remote user's thread
- Close the server of the remote session

The **`Kill`** fails if the driver stops before the command runs. Typically, failure occurs when the **`Kill`** command is entered after the Remote User panel is requested.

Note: The Remote User panel does not automatically update after a user connection is stopped.

Configuring virtual connections

The Virtual Connection Facility (VCF) component increases the number of client connections possible. When a client is active, it uses a standard “real” connection to z/OS. When the client is idle, the VCF option switches the client to a “virtual” connection. All shifts from “real” to “virtual” connections and back are transparent to the client and never interrupt a logical unit of work (LUW).

About this task

The VCF never switches from “virtual” to “real” until requests are ready for execution; that is, if LUW encapsulation is activated, the real connection is not supplied until the LUW is complete at the Data Driver.

On z/OS, the VCF maintains a thread pool, from which all active clients use a thread. When a client is idle, the z/OS thread is dropped. Use of a thread pool eliminates the overhead and time that is spent in z/OS thread creation, database thread creation, and security checking. The thread pool automatically shrinks and expands based on current activity levels for optimum performance that is balanced with low z/OS resource usage.

Set the following parameter in the AVZSIN00 configuration member:

```
“MODIFY PARM NAME(HOSTFUNCTIONALLEVEL) VALUE(x)”
```

Parameter	Description	Valid values
HOSTFUNCTIONALLEVEL	Shows what level of code the host is running. This value is passed back to the client so that the client knows what host capabilities are usable. This parameter cannot be set and is intended for technical support use only.	4 (default)

Terminating a user connection

In the Remote User program, the **Kill** command is used to terminate a remote user’s connection with the Data Virtualization Manager server. This command closes the TCP/IP session along with the driver. When a remote user’s connection is terminated, the task supporting the remote client fails with an X '222' abend. A reason code is not associated with this event.

Before you begin

The **Kill** command can only be issued by a user with authorization to do so. Authorization is granted in either of the following cases:

- When the user is granted UPDATE authority to the USERS resource.
- When the user ID attempting to kill the connection is the same as the user ID of the driver being killed. In this case, UPDATE authority is not checked.

About this task

The Server Trace program shows the following information:

- Authorization request for the kill operation
- Abend of the remote user’s thread
- Close the server of the remote session

The **Kill** fails if the driver stops before the command runs. Typically, failure occurs when the **Kill** command is entered after the Remote User panel is requested.

Note: The Remote User panel does not automatically update after a user connection is stopped.

Using CPU time limits

Data Virtualization Manager server supports an internal CPU time limit for Data Virtualization Manager client and an external CPU time limit for all clients.

Setting an internal CPU time limit for Clients

Data Virtualization Manager server provides an internal CPU time limit mechanism that limits the amount of CPU time a Client can use before it is disconnected from the host. The limit ensures that a remote Client connection does not continue by using CPU time even after the client becomes hung.

Note: The limit applies to every session and is reset each time a new session starts.

If a Client connection exceeds the CPU time limit, Data Virtualization Manager server cancels the connection and issues a message to the client and to the Server Trace log.

The time limit mechanism is activated only after a unit of work is received from the Client. It only monitors connections that are made to DB2.

Note: The internal CPU time limit mechanism does not detect time out conditions and does not stop runaway queries.

The initial time limit value is obtained from the ACF2 lid control block.

Setting an external CPU time limit for All Clients

Data Virtualization Manager server provides an external CPU time limit mechanism that limits the amount of CPU time a client can use before it is disconnected from the host, avoiding runaway queries and other CPU loops.

Note: The limit applies to every session and is reset each time a new session starts.

This mechanism involves the following limits:

- **Warning Limit.** When the warning limit is exceeded, the mechanism writes a warning message to hardcopy by identifying the user who exceeded the limit. The format of this message is:

  ```
  AVS4325H CPU TIME LIMIT EXCEEDED FOR userid FROM TCP/IP/LU x.x NODE
  name/IP address in dot notation PLAN plan name CNID connect id TP program name
  ```

- **Error Limit.** When the error limit is exceeded, the mechanism writes an error message to hardcopy by identifying the user who exceeded the limit. The format of this message is:

  ```
  AVS4326S CPU TIME LIMIT EXCEEDED FOR userid FROM TCP/IP/LU x.x NODE
  name/IP address in dot notation PLAN plan name CNID connect id TP program name
  ```

- **Failure Limit.** When the failure limit is exceeded, the application thread is terminated with an X‘522’ abend.

Note: The client does not receive a message that describes why the connection was terminated; a TCP/IP I/O error occurs when the user tries to perform the next operation.

Using wait time for all clients

Data Virtualization Manager server provides an external wait time limit mechanism that limits the amount of time a connection can remain inactive.

The external wait time limit mechanism involves the following limits:

- **Warning Limit.** When the warning limit is exceeded, the mechanism writes a warning message to hardcopy by identifying the user who exceeded the limit. The format of this message is:

  ```
  SDB4325H WAIT TIME LIMIT EXCEEDED FOR userid FROM TCP/IP/LU x.x NODE
  name/IP address in dot notation PLAN plan name CNID connect id TP program name
  ```

- **Error Limit.** When the error limit is exceeded, the mechanism writes an error message to hardcopy by identifying the user who exceeded the limit. The format of this message is:

```
SDB4326S WAIT TIME LIMIT EXCEEDED FOR userid FROM TCP/IP/LU x.x NODE
name/IP address in dot notation PLAN plan name CNID connect id TP program name
```

- **The Failure Limit.** When the failure limit is exceeded, the application thread is terminated with an X‘522’ abend. A message is sent to the client by indicating that the connection was terminated.

  Note: The client does not receive a message that describes why the connection was terminated; a TCP/IP I/O error occurs when the user tries to perform the next operation.

Detecting session failures

The Data Virtualization Manager server can also detect session failures while processing is active. This means that if a user submits a long running SQL statement or RPC and then kills the application (or reboots the system), the server detects that the session is gone and kills the SQL/RPC as soon as the session failure is known to the host.

If the application is terminated by using Task Manager (or the UNIX equivalent), the host processing terminates in a few seconds. (The default is 15 seconds.) If the client system is rebooted or part of the network fails, the host does not know about the failure until the KEEPALIVE (TCP/IP parameter) timeout occurs. The KEEPALIVE timer is usually set to 20 minutes, but it can be less or more.

Limiting the number of Data Virtualization Manager server user connections

At any one time, the number of users who are logged on the Data Virtualization Manager server cannot exceed the maximum number of users for which the Data Virtualization Manager serveris licensed. If a user tries to log on after the maximum number of users is reached, the Data Virtualization Manager server rejects the user or places the user in a queue until another user logs off, depending on the configuration of the Data Virtualization Manager server.

You can configure how you want the Data Virtualization Manager server to handle connections that exceed the licensed number of users in the following ways:

- “Using started task parameters”
- “Using the Event Facility”

Using started task parameters

Rejecting connections

To reject connections when the allowed number are exceeded, use the MODIFY PARM command in the IBM Data Virtualization Manager for z/OS Initialization EXEC, AVZSIN00, to set the CONCURRENTMX parameter:

```
"MODIFY PARM NAME(CONCURRENTMX) VALUE(xxxx)"
```

where *xxxx* is the maximum number of concurrent DB2 users. This value is a number 0 - 2000. When this value is reached, the Data Virtualization Manager server rejects further connections and returns an error message to the client.

Queuing connections

To queue connections, use the MODIFY PARM command in the IBM Data Virtualization Manager for z/OS configuration member, AVZSIN00, to set the REUSETHREADS and TARGETTHREADCOUNT parameters:

```
"MODIFY PARM NAME(REUSETHREADS) VALUE(YES)"
"MODIFY PARM NAME(TARGETTHREADCOUNT) VALUE(xxxx)"
```

where:

`PARM NAME(REUSETHREADS)` controls whether threads are reused. If this flag is set, each thread is reused a number of times if possible. If this flag is not set, a new thread is always created for each new inbound session. Thread reuse may reduce CPU resource utilization considerably when DB2 threads are used frequently and/or client user IDs are cached and reused for persistent session support. Set to YES.

`PARM NAME(TARGETTHREADCOUNT)` limits the total number of ODBC and JDBC transaction processing threads allowed in the system. The system dynamically attaches up to this number of subtasks during product execution to handle requests that arrive on TCP/IP and MQ/Series sessions. The value of this parameter can be a number from 1 to 1000.

Connections that exceed the TARGETTHREADCOUNT value queues and waits indefinitely for a new connection to become available. When a connection is released, the new connection is allowed to connect. This support typically works best with applications that use coded logic to connect and reconnect frequently based on the work performed, rather than allowing idle connections to remain. This also works with IBM Data Virtualization Manager for z/OS's Virtual Connection Facility support, which controls connections that are based on the units of work.

Using the Event Facility

Using the Event Facility, you can set specific times for connection limits by modifying Data Virtualization Manager server parameters by using the SEF /*CMD rules. These rules can be run by z/OS automatically to run an SEF program in IBM Data Virtualization Manager for z/OS to set parameters to the specified values.

You can also write SEF /*ATH rules that run at each logon to limit a user to a specific number of connections. This prevents a single workstation from connecting to Data Virtualization Manager server multiple times without disconnecting and closing old connections.

Chapter 10. Distributed transactions

Two-phase commit support is a feature of IBM Data Virtualization Manager for z/OS Enterprise Transactions. IBM Data Virtualization Manager for z/OS Enterprise Transactions support various two-phase commit protocols, with support for IBM's z/OS Recovery Resource Management Services (RRMS). This support includes the use of the Resource Recovery Services attachment facility (RRSAF).

Distributed transactions use a two-phase commit protocol to synchronize related pieces of work that take place in different processes or in different data sources.

The protocol guarantees that the work is either successfully completed by all the processes or not performed at all. The goal is to ensure that each participant in a transaction takes the same action (both are committed, or both are rolled back).

After all of the work of the transaction is complete, the application attempts to commit the work by invoking the two-phase commit protocol.

A common example for the use of two-phase commit is a banking application in which a customer wants to transfer a certain amount of money from a savings account to a checking account. Either both updates must be made or neither of them should be made. The protected resource, or in this case the amount that is being transferred, must hold its integrity in case of hardware or software failures, communication failures, human errors, or a catastrophe.

Recoverable Resource Management Services (RRMS) and the two-phase commit

The IBM Data Virtualization Manager for z/OS product on z/OS implements two-phase commit through integration with IBM's z/OS Recoverable Resource Management Services (RRMS). RRMS enables transactional access to a diverse set of resources on z/OS. RRMS also enables transactional resources to access resources outside of z/OS through distributed transaction coordinators.

With the IBM Data Virtualization Manager for z/OS support of RRMS and distributed transactions, transactions originating from the Data Drivers can provide updates to:

- DB2 on the mainframe side.
- Other z/OS data sources and transactional resources, including IMS and CICS.
- Other data sources off the mainframe.

IBM Data Virtualization Manager for z/OS on the z/OS side functions as a local transaction coordinator, communicating with the client side transaction manager (typically Microsoft MTS or BEA Tuxedo) to allow distributed transactions using the two-phase commit protocol.

Enterprise transactions for DB2

Make sure that z/OS Resource Recovery Services (RRS) is configured and running. If RRS is not running, all RRSAF requests are rejected by DB2.

- **Environment Requirements**

  Determine whether you are using two-phase commit with only DB2, or if you will also be using two-phase commit with IMS or CICS. Each subsystem has certain configuration requirements. Since two-phase commit is an overhead, you want to only enable what is necessary for the subsystem you are using. Everything else should be disabled.
- **The RRSAF Component**

  The Resource Recovery Services attachment facility (RRSAF) is a component of RRMS that allows a z/OS address space to connect to DB2. It works with RRMS, allowing updates to a DB2 system to become part of a transaction that is managed by RRMS. RRSAF also enables two-phase commit to be used by CICS/TS and IMS/TM to coordinate updates to local data sources on the z/OS operating system and to coordinate updates between two distributed data sources, such as Oracle and Sybase.

- **The CAF Component**

  Before RRSAF, call attachment facility (CAF) was the primary method that is provided by DB2 for a non-IBM address space to connect to DB2. CAF is still supported by DB2 and IBM Data Virtualization Manager for z/OS; however, it does not provide support for distributed transactions. CAF does not provide a good facility for re-using connections, and it does not provide adequate security for most customers. RRSAF provides that “request-level” facility.

  Use the following guidelines to help you decide whether to use RRSAF or CAF:

  – If you are using CAF and have a virtual storage shortage below the 16 MB line, try RRSAF. It uses at least 2 KB less per user than CAF, which helps reduce below-the-line virtual storage use.
  – If you want to do distributed two-phase commit transactions by using IBM Data Virtualization Manager for z/OS with DB2 as one of the data sources, you must choose RRSAF as the IBM Data Virtualization Manager for z/OS attachment method.
  – If only a small percentage of DB2 updates need two-phase commit transaction support, create a separate instance of the Data Virtualization Manager server for those updates, and use RRSAF with that copy.
  – RRSAF can also be used to improve performance by eliminating the need to have the Data Virtualization Manager server enqueue on DSNALI OPEN-THREAD operations.
  – RRSAF offers improved performance when used in conjunction with the Data Virtualization Manager server REUSETHREADS parameter. Since RRSAF allows for the use of a sign-on API, DB2 threads no longer have to be closed and reopened to correctly set the authid when the connection is obtained from the server’s pooled connections.

Configuring support for distributed DB2 transactions

Before you begin

Make sure that RRS is enabled on the mainframe.

Make sure the following required ODBC driver connect options are set:

- XAEN
- XAOP

Procedure

1. Set up security access. The Data Virtualization Manager server address space runs as an RRS Resource Manager. Edit and submit one of the following sample jobs in *hlq*.SAVZCNTL:
   - AVZRAXA for RACF security
   - AVZA2XA for CA ACF2 security
   - AVZTSXA for CA Top Secret security
2. If you did not establish a profile for controlling access from the RRS attachment, you must edit and submit one of the following sample jobs that are located in *HLQ*.SAVZCNTL:
   - AVZRADB2 for RACF security
   - AVZA2DB2 for CA ACF2 security
   - AVZTSDB2 for CA Top Secret security
3. Add the following parameters to the AVZSIN00 configuration member:

```
if 1 = 1 then
  do
    “MODIFY PARM NAME(RRS) VALUE(YES)”
    “MODIFY PARM NAME(DB2ATTACHFACILITY) VALUE(RRS)”
    “MODIFY PARM NAME(TRACEFULLRRSDATA) VALUE(NO)”
    “MODIFY PARM NAME(TRACERRSEVENTS) VALUE(YES)”
    “MODIFY PARM NAME(TRACERRSAF) VALUE(YES)”
    “MODIFY PARM NAME(RRS2PCALL) VALUE(YES)”
```

Parameter	**Description**	**Valid values**
DB2ATTACHFACILITY	Allows the user to control which mechanism to use for the DB2 interface. The options are to use call attachment facility (CAF) by using the DSNALI interface module or, the option of Resource Recovery Services attachment facility (RRSAF). This facility allows the capability of a two-phase commit through the attachment facility. Its interface routine is DSNRLI.	**CAF** Default value is CAF. **RRS**
RRS	Specifies whether to initialize RRS support.	**YES** **NO** Default value is NO.
RRS2PCALL	Specifies whether to use RRS COMMIT UR and RRS BACKOUT UR instead of SQL COMMIT and ROLLBACK.	**YES** **NO** Default value is NO.
TRACEFULLRRSDATA	Controls whether to trace the entire RRSAREA for RRS events.	**YES** The complete RRSAREA for RRS events is traced. **NO** (default) The complete RRSAREA for RRS events is not traced.
TRACERRSEVENTS	Specifies whether to trace RRS events.	**YES** Default value is YES. **NO**
TRACERRSAF	Specifies whether to trace each call to DSNRLI for an RRSAF request.	**YES** Default value is YES. **NO**

Configuring support for distributed DB2 transactions with the Microsoft Transaction Server

Before you begin

By default, if you use Microsoft Data Access Components (MDAC), OLE DB Session Pooling is enabled. To work with the ODBC Driver under MTS, you must turn off OLE DB Session Pooling for MSDASQL.

Make sure the required ODBC driver connect options are set.

If you change the Registry, it affects all other applications that are using the MSDASQL provider. To avoid this, you can also set this value in your application by adding the value “OLE DB Services=-4” in your connection string to turn off session pooling and autoenlistment. This setting turns off these properties for the OLE DB provider, and allows the pooling and autoenlistment to occur at the ODBC Driver level.

Procedure

1. Start the Registry Editor.

   ```
   regedit.exe
   ```

2. Navigate to the following key in the registry:

   ```
   HKEY_CLASSES_ROOT\CLSID\{c8b522cb-5cf3-11ce-ade5-00aa0044773d}\
   ```

3. Double-click the `OLEDB_SERVICES` value. The system displays the **Edit DWORD Value** dialog box.
4. In the **Value Data** field, type `0xfffffffc`, and click **OK**.

Enterprise transactions for CICS/TS and IMS

System Requirements for CICS/TS

- The mainframe and each CICS region must be enabled for Resource Recovery Services (RRS).
- If you are using a VSAM file in CICS, the file must be defined with a minimum `BACKOUTONLY` for the `RECOVERY` parameter in RDO. The default for this parameter is NONE. If the parameter value is changed from `NONE`, you must either restart the CICS region, or deallocate and then reallocate the data set to CICS with the `RECOVERY` parameter set to either `BACKOUTONLY` or `ALL`.
- See information about Setting the Required Parameters within CICS in the *CICS Transaction Server for z/OS CICS System Definition Guide*.
- RRMS authorized services, is supplied in the SDFHLINK library. For information about this link list library, refer to the *CICS Transaction Server for z/OS Installation Guide*.
- See information about Setting Parameters for the z/OS Syncpoint Manager in the *CICS Transaction Server for z/OS CICS External Interfaces Guide*, which is available in the IBM Knowledge Center.

To support RRS, you must ensure that CICS and the external CICS interface both use the z/OS syncpoint manager, which is a z/OS component of Recoverable Resource Management Services (RRMS). In the context of RRMS, CICS is a resource manager. The client program can issue requests to other resource managers and have resources that are owned by those resource managers who are committed in the same unit-of-recovery (UR).

These options are controlled as follows:

- By the `DPL_opts` parameter of the DPL_request.
- By the `SYNCONRETURN` option, either specified or omitted, on the `EXEC CICS LINK PROGRAM` command.

If you specify `SYNCONRETURN`, a syncpoint is taken on completion of each DPL request. If `SYNCONRETURN` is omitted, a syncpoint is taken when the client program requests it using the interfaces that are described in “Taking a Syncpoint in the Client Program” in the *CICS Transaction Server for z/OS CICS External Interfaces Guide*.

Environment Requirements

You should determine whether you use two-phase commit with only DB2, or if you also use two-phase commit with IMS or CICS. Each subsystem has certain configuration requirements. Since two-phase commit is an overhead, you want to only enable what is necessary for the subsystem you are using. Everything else should be disabled.

APPC Environment Requirements for IMS

When `SYNCLVL=SYNCPT` is specified, Advanced Program-to-Program Communication (APPC) acquires a private context on behalf of IMS. IMS provides its resource manager name to APPC in its identity call. APPC provides the private context to IMS as the message header. IMS, using this context, then assumes the role of a participant in the two-phase commit process with the syncpoint manager, RRS/MVS.

In order to use SYNCLVL=SYNCPT, you must also be sure that an APPC/MVS logstream is defined as documented in *z/OS MVS Planning: APPC/MVS Management*. Otherwise an ATB222I error occurs when the LU is used.

In addition to SYNCLVL=SYNCPT, the keyword ATNLOSS=ALL must be specified in the VTAM definition file for whichever LUs the user wishes to enable for protected conversations.

For more information, refer to the *IMS/ESA Administration Guide: Transaction Manager*.

OTMA Environment Requirements for IMS

In an OTMA (Open Transaction Manager Access) environment, OTMA is not a resource manager who is registered with RRS/MVS. The process remains an interprocess protocol between a Server (IMS) and a number of clients (application programs).

Therefore, OTMA cannot obtain a private context token to pass to IMS as APPC does. The client-driver code that uses OTMA is responsible for obtaining and owning a private context and for providing the context ID. In messages that are passed between the partners, the context ID field contains the context token (if it is a protected conversation).

When IMS finds the context ID in the message, IMS assumes the role of a participant in the two-phase commit process, as it does in the APPC environment.

For more information about these OTMA topics, refer to the *IMS/ESA Open Transaction Manager Access Guide*.

Configuring support for CICS and IMS distributed transactions

Before you begin

Data Drivers

Make sure the following required Data Drivers connect options are configured:

- XAOP

Procedure

1. Complete the following steps to grant ALTER access to the Data Virtualization Manager server address space ID for the MVSADMIN.RRS.COMMANDS RACF resource class.
   a) Enter the following command to grant access to the resource:

   ```
   permit MVSADMIN.** class(facility) ID(AVZS) access(alter)
   ```

   Where AVZS is the name of the Data Virtualization Manager server address space.
   b) Enter the following command to refresh the facility class:

   ```
   setropts refresh class(facility) raclist
   ```

2. Add the following parameters that are located in the Data Virtualization Manager configuration member, AVZSIN00:

   ```
   if 1 = 1 then
     do
       “MODIFY PARM NAME(RRS) VALUE(YES)”
       “MODIFY PARM NAME(RRSCICS) VALUE(NO)”
       “MODIFY PARM NAME(RRSIMSTM) VALUE(NO)”
       “MODIFY PARM NAME(RECTABLEENTRIES) VALUE(10000)”
       “MODIFY PARM NAME(RESOURCEMGRNAME) VALUE(NEONRMAAAAAA)”
       “MODIFY PARM NAME(TRACEFULLRRSDATA) VALUE(NO)”
       “MODIFY PARM NAME(TRACERRSEVENTS) VALUE(YES)”
   ```

   The following table lists the parameter for configuring Enterprise Transactions for CICS support:

Parameter	Description	Valid values
RRS	Specifies whether to enable RRS support.	**YES** **NO** Default value is YES.
RRSCICS (*CICS only*)	Specifies whether to enable RRS support for CICS.	**YES** **NO** Default value is NO.
RRSIMSTM (*IMS only*)	Allows the user to specify the RRS/two-phase interaction for IMS only.	**YES** **NO** Default value is NO.
RECTABLEENTRIES	Specifies the maximum number of entries in the RRS recovery table. If the maximum number of entries is exceeded, information about in-doubt transactions is lost.	400 (default) Valid values are 200 - 400.
RESOURCEMGRNAME	Specifies the unique sysplex name of the RRS resource manager, which is a server distributed syncpoint manager (SDSRM). If a default value is not specified, product initialization creates a 32-character name as follows: If the name is changed after the system is operational, in-doubt transactions from the previous run cannot be completed.	`'NEONRRS.RESOURCE.MANAGER'` • Chars 1-24: NEONRRS.RESOURCE.MANAGER • Chars 25-28: The IBM Data Virtualization Manager for z/OS subsystem name (AVZS) • Chars 29-32: System SMF ID
TRACEFULLRRSDATA	Specifies whether to trace the entire RRS work area.	**YES** **NO** (default) Trace only the amount of data that fits in a standard message block.
TRACERRSEVENTS	Specifies whether to trace RRS events.	**YES** Default value is YES. **NO**

Chapter 11. Migrating maps

Use the Map Migration utility to move your virtual table maps from a development environment to a test or production environment or from one release to another.

Before you begin

Before using the Map Migration utility, make sure that the following prerequisites have been met:

- **Data Virtualization Manager studio requirements**

  If migrating DB2 virtual tables, target systems used by each table must be defined in the target server using one of the following definitions:

  – If you want to use the same target system name, define the target system name on the target server.

  – If you want to use a different target system name, then define the new target system name, and use the `TSYS=OLD_TSYS,NEW_TSYS` parameter in the AVZGNMPM batch migration utility.

- **Data Virtualization Manager server requirements**

  Make sure that both the origin and destination servers have been started.

- **Data Virtualization Manager server security requirements**

  The following table summarizes the security permissions required to use the migration utility:

Table 68. Security permissions required to use the migration utility

	JCL library	**Map export PDS**	**Server map data set**
	The location where the JCL resides.	The PDS library to which the exported metadata objects are unloaded.	The `AVZMAPP DD` data set, which must be the first data set in the concatenation if the parameter `NEW MAP DSN` is not set.
Batch user ID	UPDATE	CREATE READ	N/A
Server user ID	N/A	UPDATE	UPDATE READ

About this task

The Map Migration utility facilitates change control of the virtual table maps. Change control is the process of moving the virtual table maps defined in a development environment to a test or production environment or from one release to another.

You can use the AVZGNMPM member located in your *hlq*.SAVZCNTL data set for migrating virtual table maps. See the AVZGNMPM member for a list of parameters available for use when migrating virtual table maps.

You can use the AVZGNMPM member to perform the following tasks:

- Migrate one or multiple virtual table maps from one server to another.
- Change the virtual table map definition using the optional parameters. See the comments in the sample job for more details.

Procedure

1. Customize the migration utility job, AVZGNMPM, for the requirements at your site.
2. Submit the AVZGNMPM batch job. Utility job AVZGNMPM extracts the contents of the maps, stores the metadata objects in the map export PDS library, and creates the batch job that is used to rebuild the maps on the target server.
3. Submit the batch JCL that was created in the previous step to rebuild the maps on the target server.

Results

The utility extracts the content of the map export PDS and rebuilds the map on the target server.

Chapter 12. Supported SQL Functions

This chapter describes the SQL functions supported in IBM® Data Virtualization Manager for z/OS.

Sample Table For Examples

This section lists the sample tables used across examples in the subsequent SQL functions sections.

Table 69. VIRTUAL TABLE - EMPL_COMP

EMPL_ID	EMPL_NAME	COMPANY	LOCATION	UPDATED_TS
1	AKHIL	TIMBER INC.	NY	2019-11-26 05:39:59.454707
2	ELICA	STANLEY INC.	TX	
3	SMITH	CHROME INC.	WA	
1919574970	KANE			

Table 70. VIRTUAL TABLE - EMPLOYEE

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					

Table 71. Virtual Table - EMP

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
DEV	33	ADMIN	RETAIL	25000	2019-11-27
WAYNE	36	ADMIN	RETAIL	45000	2018-11-01

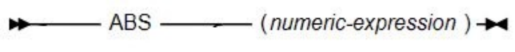
Table 72. Virtual Table - JOIN1

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
999	JOHN		SALES		16.99	
10	MIC	20	MGR	7	98357	0.00
20	WILLIAM	20	SALES	8	78171	612.45
30	MIC	38	MGR	5	77506	0.00
40	OBRIEN	38	SALES	6	78006	846.55

Table 73. Virtual Table - JOIN2						
ID	**NAME**	**DEPT**	**JOB**	**YEARS**	**SALARY**	**COMM**
0	SANDERS	20	MGR	7	18357	
1	WILLIAM	20	SALES	8	18171	612.45

ABS

The ABS function returns the absolute value of a number.



The argument must be an expression that returns a value of any built-in numeric data type.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The result of the function has the same data type and length attribute as the argument.

Notes

Syntax alternatives:
ABS should be used for conformance to the SQL standard.

Example:

The following statement returns the absolute value of the values in the *SALARY* column from the table EMPLOYEE.

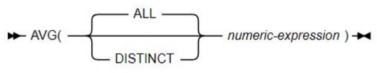
```
SELECT ABS(SALARY) FROM EMPLOYEE;
```

The above example returns the following:

```
50000
20000
5000
25000
75000
75000
```

AVG

The AVG function returns the average of a set of numbers.



The argument values can be of any built-in numeric data type, and their sum must be within the range of the data type of the result.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The data type of the result is determined as follows:

- DECFLOAT(34) if the argument is DECFLOAT(n).
- Large integer if the argument is small integer.
- Double precision floating-point if the argument is single precision floating-point.

- Otherwise, the result is the same as the data type of the argument.

If the data type of the argument values is decimal with precision p and scale s, the precision (P) and scale (S) of the result depend on p and the decimal precision option:

- If *p* is greater than 15 or the DEC31 option is in effect, P is 31 and S is `max(0,28-p+s)`.
- Otherwise, P is 15 and S is 15-p+s.

The function is applied to the set of values derived from the argument values by the elimination of null values. If DISTINCT is specified, redundant duplicate values are also eliminated.

If the type of the result is integer, the fractional part of the average is lost.

Example:

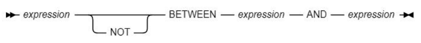
The following example returns the average of values in the column *SALARY* in the table EMPLOYEE.

```
SELECT AVG(SALARY) FROM EMPLOYEE;
```

The above example returns *49000.00000*.

BETWEEN

The BETWEEN predicate determines whether a given value lies between two other given values that are specified in ascending order.



Each of the predicate's two forms has an equivalent search condition, as shown in the following table:

Table 74. BETWEEN predicate and equivalent search conditions

BETWEEN predicate	**Equivalent search condition**
value1 BETWEEN *value2* AND \value3	*value1* >= *value2* AND *value1* <= *value3* **Note:** Might not be equivalent if *value1*, *value2*, or *value3* are columns or derived values based on columns that are not the same CCSID set because the clause is evaluated in Unicode.
value1 NOT BETWEEN *value2* AND *value3* or, equivalently: NOT(*value1* BETWEEN *value2* AND *value3*)	*value1* < *value2* OR *value1* > *value3* **Note:** Might not be equivalent if *value1*, *value2*, or *value3* are columns or derived values based on columns that are not the same CCSID set because the clause is evaluated in Unicode.

If the operands include a mixture of datetime values and valid string representations of datetime values, all values are converted to the data type of the datetime operand.

Example:

```
A BETWEEN B AND C
```

The following example returns the record where the age of an employee in the table EMPLOYEE is *>50*.

```
SELECT EMPL_NAME,AGE FROM EMPLOYEE
WHERE AGE >50
```

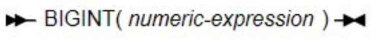
The above example returns the following.

Table 75. Between

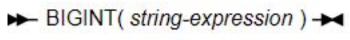
EMPL_NAME	AGE
BUMRAH	53
NEWTON	53

BIGINT

The BIGINT function returns a big integer representation of either a number or a character or graphic string representation of a number.



Numeric to Big Integer



numeric-expression
An expression that returns a value of any built-in numeric data type.

The result is the same number that would occur if the argument were assigned to a big integer column or a variable. If the whole part of the argument is not within the range of big integers, an error is returned. The fractional part of the argument is truncated.

String to Big Integer

string-expression
An expression that returns a value of a character or graphic string (except a CLOB and DBCLOB) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number.

The result is the same number that would result from CAST(`string-expression AS BIGINT`). Leading and trailing blanks are eliminated and the resulting string must conform to the rules for forming an integer constant. If the whole part of the argument is not within the range of big integers, an error is returned.

The result of the function is a big integer.

To increase the portability of applications, use the CAST specification.

Example 1:

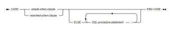
The following example returns 7 from the table EMPLOYEE:

```
SELECT BIGINT(COUNT(*)) FROM EMPLOYEE;
```

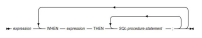
CASE

The CASE statement selects an execution path based on the evaluation of one or more conditions. A CASE statement operates in the same way as a CASE expression.

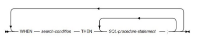
Syntax



simple-when-clause:



searched-when-clause:



Description

CASE
: Begins a case-expression.

simple-when-clause
: Specifies the expression prior to the first WHEN keyword that is tested for equality with the value of each expression that follows the WHEN keyword, and the result to be executed when those expressions are equal. If the comparison is true, the THEN statement is executed. If the result is unknown or false, processing continues to the next expression or the ELSE statement.

  The data type of the expression prior to the first WHEN keyword must be comparable to the data types of each expression that follows the WHEN keywords.

searched-when-clause
: Specifies the search-condition that is applied to each row or group of table data presented for evaluation, and the result when that condition is true. search-condition cannot contain a fullselect. If the search condition is true, the THEN statement is executed. If the condition is unknown or false, processing continues to the next search condition or the ELSE statement.

SQL-procedure-statement
: Specifies a statement that follows the THEN and ELSE keyword. The statement specifies the result of a searched-when-clause or a simple-when-clause that is true, or the result if no case is true.

search-condition
: Specifies a condition that is true, false, or unknown about a row or group of table data.

ELSE SQL-procedure-statement
: If none of the conditions specified in the simple-when-clause or searched-when-clause are true, the statements in the else-clause are executed.

  If none of the conditions specified in the WHEN clause are true and an ELSE clause is not specified, an error is returned at run time, and the execution of the CASE statement is terminated.

END CASE
: Ends a case-statement.

Note:

If none of the conditions specified in the WHEN clause are true and an ELSE clause is not specified, an error is returned at run time, and the execution of the CASE statement is terminated.

CASE statements that use a simple case statement WHEN clause can be nested up to three levels. CASE statements that use a searched statement WHEN clause have no limit to the number of nesting levels.

Examples:

The following example assigns the experience level of *1* to employees with age *<40* and the experience level of *2* to employees with age *>40* from the table EMPLOYEE.

```
SELECT EMPL_NAME,
CASE
WHEN AGE > 40 THEN 'LEVEL 2'
WHEN AGE < 40 THEN 'LEVEL 1'
END AS EXP_LEVEL,
FROM EMPLOYEE;
```

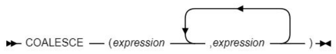
The above example returns the following:

Table 76. Case

EMPL_NAME	EXP_LEVEL
BUMRAH	LEVEL 2
CEASAR	LEVEL 2
ELICA	LEVEL 1
DEV	LEVEL 1
NEWTON	LEVEL 2
SAMUEL	LEVEL 2
ELICA	

COALESCE

The function COALESCE returns the value of the first non-null expression.



The arguments can be of either a built-in or user-defined data type.

The COALESCE function cannot be used as a source function when creating a user-defined function.

The arguments are evaluated in the order in which they are specified, and the result of the function is the first argument that is not null.

The selected argument is converted, if necessary, to the attributes of the result. If the COALESCE function has more than two arguments, the rules are applied to the first two arguments to determine a candidate result type. The rules are then applied to that candidate result type and the third argument to determine another candidate result type. This process continues until all arguments are analyzed and the final result type is determined.

If there are any mixed character string or graphic string and numeric arguments, the string value is implicitly cast to a DECFLOAT(34) value.

The COALESCE function can also handle a subset of the functions provided by CASE expressions. The result of using `COALESCE(e1,e2)` is the same as using the expression:

```
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END
```

VALUE can be specified as a synonym for COALESCE.

Example 1

The following example returns the value of the first nonnull expression between *DATEOFJOIN* and *CURRENT_DATE* from the table EMPLOYEE.

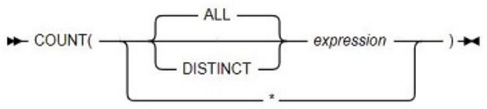
```
SELECT COALESCE(DATEOFJOIN, CURRENT_DATE) AS JOINDATE FROM EMPLOYEE;
```

The above example returns the following.

```
JOINDATE
2019-11-25
2020-01-06
2019-10-10
2019-11-27
2019-11-25
2019-11-25
2020-01-06
```

COUNT

The COUNT function returns the number of rows or values in a set of rows or values.



The argument values can be of any built-in data type other than a BLOB, CLOB, DBCLOB, or XML.

If the argument of `COUNT(*)` is a set of rows, the result would be the number of rows in the set. Any row that includes only null values is included in the count.

If the argument of `COUNT(expression)` or `COUNT(ALL expression)` is a set of values, the function is applied to the set of values derived from the argument values by the elimination of null values. The result is the number of nonnull values in the set, including duplicates.

If the argument of `COUNT(DISTINCT expression)` is a set of values, the function is applied to the set of values derived from the argument values by the elimination of null values and redundant duplicate values. The result is the number of different nonnull values in the set.

Example

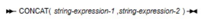
The following statement counts the number of rows from the table EMPLOYEE.

```
SELECT COUNT(*) FROM EMPLOYEE
```

The above example returns *7*.

CONCAT

The function CONCAT combines two compatible string arguments.



Either argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

The result of the function is a string that consists of the first string followed by the second string.

The CONCAT function is identical to the CONCAT operator.

Example:

The following example combines strings available in the columns *EMPL_NAME* and *DEPT* from the table EMPLOYEE.

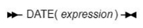
```
SELECT CONCAT(EMPL_NAME , ' ', DEPT) AS EMPL_DEPT FROM EMPLOYEE
```

The above example returns the following.

```
EMPL_DEPT
BUMRAH SALES
CEASAR MARKETING
ELICA CUST SUPR
DEV ADMIN
NEWTON DATA
SAMUEL IT
```

DATE

The DATE function returns a date that is derived from a value.



The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or any numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and it must have one of the following values:
  - A valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.
  - A character or graphic string with an actual length of 7 that represents a valid date in the form yyyynnn, where yyyy are digits denoting a year and nnn are digits between 001 and 366 denoting a day of that year.
- If expression is a number, it must be greater than or equal to one and less than or equal to 3652059.

If expression is not a DATE value, expression is cast as follows:

- – If expression is a TIMESTAMP WITH TIME ZONE value, expression is cast to TIMESTAMP WITHOUT TIME ZONE, with the same precision as expression.
  – If expression is a string, expression is cast to DATE.

The result of the function is a date.

The other rules depend on the data type of the argument:

- **If the argument is a timestamp**, the result is the date part of the timestamp.
- **If the argument is a date**, the result is that date.
- **If the argument is a number**, the result is the date that is n-1 days after January 1, 0001, where n is the integral part of the number.
- **If the argument is a string**, the result is the date that is represented by the string. If the string contains a time zone, the time zone is ignored. If the CCSID of the string is not the same as the corresponding default CCSID at the server, the string is first converted to that CCSID.

  The result CCSID is the appropriate CCSID of the argument encoding scheme and the result subtype is the appropriate subtype of the CCSID.

Example 1:

The following example selects records of employees who joined later than *2019-11-10* from the table EMPLOYEE.

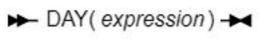
```
SELECT * FROM EMPLOYEE WHERE DATEOFJOIN > '2019-11-10'
```

The above example returns the following.

```
EMPL_NAME, AGE, DEPT, INDUSTRY, SALARY, DATEOFJOIN
BUMRAH , 53 , SALES , RETAIL , 50000 , 2019-11-25
DEV , 33 , ADMIN , RETAIL , 25000 , 2019-11-27
NEWTON , 53 , DATA , RETAIL , 75000 , 2019-11-25
SAMUEL , 43 , IT , BANKING , 75000 , 2019-11-25
```

DAY

The DAY function returns the day part in the given argument.



The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or any numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.
- If expression is a number, it must be a date duration or a timestamp duration.

If expression is a timestamp with a time zone value, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules for the function depend on the data type of the argument:

- **If the argument is a date, timestamp, or string representation of either**, the result is the day part of the value, which is an integer between 1 and 31.
- **If the argument is a date duration or timestamp duration**, the result is the day part of the value, which is an integer between -99 and 99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example:

The following example returns the day part in the given string *DATEOFJOIN* from the table EMPLOYEE:

```
SELECT DAY(DATEOFJOIN) FROM EMPLOYEE
```

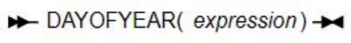
The above example returns the following.

```
25

10
27
25
25
```

DAYOFYEAR

The DAYOFYEAR function returns an integer, in the range of 1 to 366, that represents the day of the year, where 1 is January 1.



The argument must be an expression that returns a value of one of the following built-in data types: a date, a timestamp, a character string, or a graphic string.

If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length that is not greater than 255 bytes.

If expression is a timestamp with a time zone value, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

Example 1:

The following example selects the day part from the given value *DATEOFJOIN* from the table EMPLOYEE:

```
SELECT DAYOFYEAR(DATEOFJOIN) FROM EMPLOYEE
```

The above example returns the following.

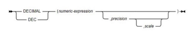
```
329
```

```
283
331
329
329
```

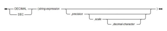
DECIMAL

The DECIMAL function returns a decimal representation of either a number or a character-string or graphic-string representation of a number, an integer, or a decimal number.

Numeric to Decimal:



String to Decimal:



Numeric to decimal

numeric-expression
An expression that returns a value of any built-in numeric data type.

precision
An integer constant with a value greater than or equal to 1 and less than or equal to 31.

The default for precision depends on the data type of the numeric-expression:

- 5 for small integer
- 11 for large integer
- 19 for big integer
- 15 for floating point or decimal
- 31 for decimal floating point

scale
An integer constant that is greater than or equal to zero and less than or equal to precision. The value specifies the scale of the result. The default value is 0.

The result of the function is the same number that would occur if the argument were assigned to a decimal column or variable with precision p and scale s, where p and s are specified by the second and third arguments. An error occurs if the number of significant digits required to represent the whole part of the number is greater than `p-s`.

String to decimal

string-expression
An expression that returns a value of a character or graphic string (except a CLOB or DBCLOB) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number. Leading and trailing blanks are removed from the string, and the resulting substring must conform to the rules for forming a valid string representation of an SQL integer or decimal constant.

precision
An integer constant with a value in the range 1 to 31. The value of this second argument specifies the precision of the result. If not specified, the default is 15.

scale
An integer constant that is greater than or equal to zero and less than or equal to precision. The value specifies the scale of the result. The default value is 0.

decimal-character
A single-byte character constant used to delimit the decimal digits in string-expression from the whole part of the number. The character cannot be a digit, plus (+), minus (-), or blank. The default

value is period (.) or comma (,); the default value cannot be used in string-expression if a different value for decimal-character is specified.

Digits are truncated from the end of the decimal number if the number of digits to the right of the decimal separator character is greater than the scale s. An error is returned if the number of significant digits to the left of the decimal character (the whole part of the number) in string-expression is greater than `p-s`.

The result of the function is a decimal number with precision of p and scale of s, where p and s are the second and third arguments.

Notes

Syntax alternatives:
To increase the portability of applications when the precision is specified, use the CAST specification.

Example
Represent the average salary of the employees in the table EMPLOYEE as an 8-digit decimal number with two of these digits to the right of the decimal point.

```
SELECT DECIMAL(AVG(SALARY),8,2)
     FROM EMPLOYEE;
```

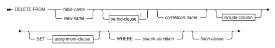
The above example returns *49000.00*

DELETE

The DELETE statement deletes rows from a table or a view. The table or view can be at the current server or DVM server with which the current server can establish a connection. There are two forms of this statement:

- The searched DELETE form is used to delete one or more rows, optionally determined by a search condition.
- The positioned DELETE form specifies that one or more rows corresponding to the current cursor position are to be deleted.

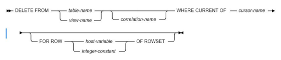
searched delete:



Note:

- If the period-clause is specified, the fetch-clause must not be specified.
- The same clause must not be specified more than one time.

positioned delete:



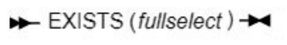
Example:

The following example delete a record that matches the condition *EMPL_NAME = 'DAVE'* from the table EMPLOYEE.

```
DELETE FROM EMPLOYEE
WHERE EMPL_NAME = 'DAVE'
```

EXISTS

The EXISTS predicate tests for the existence of certain rows. The fullselect can specify any number of columns, and can result in true or false.



Notes:

- The outer SELECT list of fullselect must not contain an array value.

The result of the EXISTS predicate:

- Is true only if the number of rows that is specified by the fullselect is not zero.
- Is false only if the number of rows specified by the fullselect is zero.
- Cannot be unknown.

The SELECT clause in the fullselect can specify any number of columns because the values returned by the fullselect are ignored. For convenience, use: `SELECT *`

Unlike NULL, LIKE, and IN predicates, the EXISTS predicate has no form that contains the word NOT. To negate an EXISTS predicate, precede it with the logical operator NOT, as : `NOT EXISTS (fullselect)`

The result is then false if the EXISTS predicate is true, and true if the predicate is false. Here, NOT is a logical operator and not a part of the predicate.

Example 1:

The following example compares the values in the column *EMPL_NAME* of the table EMPLOYEE with the values in the column *EMPL_NAME* of the table EMPL_COMP and returns the record that has a similar name with age *>20*.

```
SELECT T1.EMPL_ID, T1.EMPL_NAME FROM EMPL_COMP T1
WHERE EXISTS
SELECT T2.AGE FROM EMPLOYEE T2  WHERE T1.EMPL_NAME = T2.EMPL_NAME AND T2.AGE > 20)
```

The above example returns the following.

Table 77. Exists

EMPL_ID	**EMPL_NAME**
2	ELICA

FULL OUTER JOIN

The FULL OUTER JOIN clause results in the inclusion of rows from two tables. If a value is missing when rows are joined, that value is null in the resultant table.

The join condition for a full outer join must be a search condition that compares two columns. The predicates of the search condition can be combined only with AND. Each predicate must have the form 'expression = expression'.

Example:

The following query performs a full outer join of the EMPLOYEE and EMPL_COMP tables:

```
SELECT A.EMPL_NAME, B.EMPL_ID, A.DEPT, A.INDUSTRY, B.COMPANY, B.LOCATION
FROM EMPLOYEE A
FULL OUTER JOIN EMPL_COMP B
ON
A.EMPL_NAME = B.EMPL_NAME
```

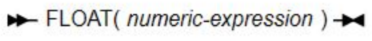
The result table looks like this:

Table 78. Full Outer Join

EMPL_NAME	EMPL_ID	DEPT	INDUSTRY	COMPANY	LOCATION
	1			TIMBER INC.	
BUMRAH		SALES	RETAIL		
CEASAR		MARKETING	RETAIL		
DEV		ADMIN	RETAIL		
ELICA	2	CUST SUPR	RETAIL	STANLEY INC.	TX
ELICA	2			STANLEY INC.	TX
NEWTON		DATA	RETAIL		

FLOAT

The FLOAT function returns a floating-point representation of either a number or a string representation of a number.



Notes

Syntax alternatives:
FLOAT is a synonym for DOUBLE_PRECISION or DOUBLE.

Example:

The following example returns the floating point representation of the values in the *AGE* from the table EMPLOYEE.

```
SELECT FLOAT(AGE) FROM EMPLOYEE
```

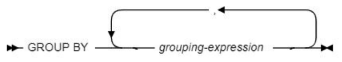
The above example returns the following.

```
53.0
43.0
23.0
33.0
53.0
43.0
```

GROUP-BY

The GROUP BY clause specifies a result table that consists of a grouping of the rows of intermediate result table that is the result of the previous clause.

group-by-clause



In its simplest form, a GROUP BY clause contains a grouping-expression.

grouping-expression
A grouping-expression is an expression that defines the grouping of R. The following restrictions apply to grouping-expression:

- If grouping-expression is a single column, the column name must unambiguously identify a column of R.
- The result of grouping-expression cannot be a LOB data type (or a distinct type that is based on a LOB) or an XML data type.
- grouping-expression cannot include any of the following items:
  - A correlated column
  - A host variable
  - An aggregate function
  - Any function or expression that is not deterministic or that is defined to have an external action
  - A scalar fullselect
  - A CASE expression whose searched-when-clause contains a quantified predicate, an IN predicate using a fullselect, or an EXISTS predicate

The result of GROUP BY is a set of groups of rows. In each group of more than one row, all values of each grouping-expression are equal, and all rows with the same set of values of the grouping-expression are in the same group.

If a grouping-expression contains DECFLOAT values, the DECFLOAT values with the same value will be in the same group. But the number of digits returned for each group is unpredictable.

Because every row of a group contains the same value of any grouping-expression, a grouping-expression can be used in a search condition in a HAVING clause or an expression in a SELECT clause, or in a sort-key-expression of an ORDER BY clause. In each case, the reference specifies only one value for each group. For example, if grouping-expression is col1+col2, col1+col2+3 would be an allowed expression in the select list. Associative rules for expressions do not allow the similar expression of 3+col1+col2, unless parentheses are used to ensure that the corresponding expression is evaluated in the same order. Thus, 3+(col1+col2) would also be allowed in the select list. If the concatenation operator is used, grouping-expression must be used exactly as the expression was specified in the select list.

If a grouping-expression contains varying-length strings with trailing blanks, the values in the group can differ in the number of trailing blanks and might not all have the same length. In that case, a reference to grouping-expression still specifies only one value for each group, but the value for a group is chosen arbitrarily from the available set of values. Thus, the actual length of the result value is unpredictable.

Example:

The following example groups the record by *INDUSTRY* from the table EMPLOYEE.

```
SELECT COUNT(DEPT), INDUSTRY FROM EMPLOYEE
GROUP BY INDUSTRY
```

The above example returns the following.

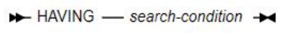
Table 79. Group-By

	INDUSTRY
1	BANKING
5	RETAIL

HAVING

The HAVING clause specifies a result table that consists of those groups of the intermediate result table for which the search-condition is true. The intermediate result table is the result of the previous clause. If this clause is not GROUP BY, the intermediate result table is considered a single group with no grouping columns of the previous clause of the subselect.

having-clause



Each column-name in search-condition must be one of the following:

- Unambiguously identify a grouping column of the intermediate result table
- Be specified within an aggregate function
- Be a correlated reference. A column-name is a correlated reference if it identifies a column of a table, view, common-table-expression, or nested-table-expression that is identified in an outer subselect

A group of the intermediate result table to which the search condition is applied supplies the argument for each function in the search condition, except for any function whose argument is a correlated reference.

If the search condition contains a subquery, the subquery can be thought of as being executed each time the search condition is applied to a group of the intermediate result table, and the results used in applying the search condition. In actuality, the subquery is executed for each group only if it contains a correlated reference.

A correlated reference to a group of the intermediate result table must either identify a grouping column or be contained within an aggregate function.

When HAVING is used without GROUP BY, any expression or column name in the select list must appear within an aggregate function.

Example:

The following example groups the records by *INDUSTRY* column from the table EMPLOYEE and returns the records with count *>= 1* .

```
SELECT COUNT(DEPT) AS COUNT, INDUSTRY
FROM EMPLOYEE
GROUP BY INDUSTRY
HAVING COUNT(DEPT) >= 1
```

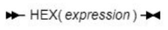
The above example returns the following.

Table 80. Having

COUNT	**INDUSTRY**
1	BANKING
5	RETAIL

HEX

The HEX function returns a hexadecimal representation of a value.



The argument must be an expression that returns a value of any built-in data type that is not XML. A character or binary string must not have a maximum length greater than 16352. A graphic string must not have a maximum length greater than 8176.

The result of the function is a character string.

The result is a string of hexadecimal digits. The first two represent the first byte of the argument, the next two represent the second byte of the argument, and so forth. If the argument is a datetime value, the result is the hexadecimal representation of the internal form of the argument.

If the argument is a fixed-length string and the length of the result is less than 255, the result is a fixed-length string. Otherwise, the result is a varying-length string with a length attribute that depends on the following considerations:

- **If the argument is not a varying-length string**, the length attribute of the result string is the same as the length of the result.
- **If the argument is a varying-length character or binary string**, the length attribute of the result string is twice the length attribute of the argument.
- **If the argument is a varying-length graphic string**, the length attribute of the result string is four times the length attribute of the argument.

If expression returns string data, the CCSID of the result is the SBCS CCSID that corresponds to the CCSID of expression. Otherwise, the CCSID of the result is determined from the context in which the function was invoked.

If the argument is a graphic string, the length of the result is four times the maximum length of the argument. Otherwise, the length of the result is twice the (maximum) length of the argument.

Example:

The following example returns the hexadecimal value of the column *AGE* from the table EMPLOYEE.

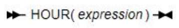
```
SELECT HEX(AGE) FROM EMPLOYEE
```

The above example returns the following:

```
00000035
0000002B
00000017
00000021
00000035
0000002B
```

HOUR

The HOUR function returns the hour part of the given argument.



The argument must be an expression that returns a value of one of the following built-in data types: a time, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a time or timestamp duration.

If expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a time, timestamp, or string representation of either**, the result is the hour part of the value, which is an integer between 1 and 24.
- **If the argument is a time duration or timestamp duration**, the result is the hour part of the value, which is an integer between -99 and +99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example:

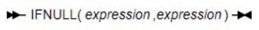
The following example returns the hour part in the string *UPDATED_TS* from the table EMPL_COMP.

```
SELECT HOUR(UPDATED_TS) FROM EMPL_COMP
```

The above example returns *5*.

IFNULL

The IFNULL function returns the first nonnull expression.



IFNULL is identical to the COALESCE scalar function except that IFNULL is limited to two arguments instead of multiple arguments.

Example:

The following example returns 0 if the value in the *SALARY* column from the table EMPLOYEE is NULL.

```
SELECT EMPL_NAME, IFNULL(SALARY,0)
     FROM EMPLOYEE;
```

The above example returns the following.

```
EMPL_NAME,
BUMRAH , 50000
CEASAR , 20000
ELICA , 0
DEV , 25000
NEWTON , 75000
SAMUEL , 75000
ELICA , 0
```

INNER JOIN

You can use an inner join in a SELECT statement to retrieve only the rows that satisfy the join conditions on every specified table.

You can request an inner join, by running a SELECT statement in which you specify the tables that you want to join the FROM clause and specify a WHERE clause or an ON clause to indicate the join condition. The join condition can be any simple or compound search condition that does not contain a subquery reference.

In the simplest type of inner join, the join condition is *column1=column2*.

Example:

The following example joins the records with similar employee name from the tables EMPLOYEE and EMPL_COMP.

```
SELECT A.EMPL_NAME, B.EMPL_ID, A.DEPT, A.INDUSTRY, B.COMPANY, B.LOCATION
FROM EMPLOYEE A
INNER JOIN EMPL_COMP B
ON
A.EMPL_NAME = B.EMPL_NAME "
```

The above example returns the following.

Table 81. Inner Join

EMPL_NAME	**EMPL_ID**	**DEPT**	**INDUSTRY**	**COMPANY**	**LOCATION**
ELICA	2	CUST SUPER	RETAIL	STANLEY INC.	TX
ELICA	2			STANLEY INC.	TX

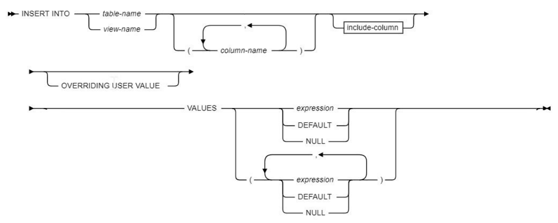
INSERT

The INSERT statement inserts rows into a table or view. The table or view can be at the current server or any DVM server with which the current server can establish a connection.

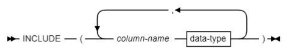
The INSERT via VALUES form is used to insert a single row into the table or view using the values provided or referenced.

The owner of a view, unlike the owner of a table, might not have INSERT authority on the view (or can have INSERT authority without being able to grant it to others). The nature of the view itself can preclude its use for INSERT.

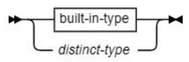
Syntax



include-column:



data-type:



Example:

The example inserts the values *(0003,'SMITH','CHROME INC.','WA')* to the table EMPLOYEE:

```
INSERT INTO EMPL_COMP (EMPL_ID, EMPL_NAME, COMPANY, LOCATION) VALUES(0003,'SMITH',
'CHROME INC.','WA');
```

The following is the updated table after the record insertion.

Table 82. Insert

EMPL_ID	EMPL_NAME	COMPANY	LOCATION	UPDATED_TS
1	AKHIL	TIMBER INC.	NY	2019-11-26 05:39:59.454707
2	ELICA	STANLEY INC.	TX	
3	SMITH	CHROME INC.	WA	

LARGE INTEGER (INTEGER)

A *large integer* is a binary integer with a precision of 31 bits.

The range of large integers is -2147483648 to +2147483647.

Example:

The following example selects the values in the *SALARY* column of the table EMPLOYEE as an integer.

```
SELECT CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

The above example returns the following.

```
50000
20000
```

```
25000
75000
75000
```

LEFT OUTER JOIN

The LEFT OUTER JOIN clause lists rows from the left table even if there are no matching rows on right table.

As in an inner join, the join condition of a left outer join can be any simple or compound search condition that does not contain a subquery reference

Example:

Consider the following example using the records in the table EMPLOYEE:

```
SELECT A.EMPL_NAME, B.EMPL_ID, A.DEPT, A.INDUSTRY, B.COMPANY, B.LOCATION
FROM EMPLOYEE A
LEFT OUTER JOIN EMPL_COMP B
ON
A.EMPL_NAME = B.EMPL_NAME
```

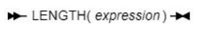
The result table looks like the following example:

Table 83. Left Outer Join

EMPL_NAME	EMPL_ID	DEPT	INDUSTRY	COMPANY	LOCATION
BUMRAH		SALES	RETAIL		
CEASAR		MARKETING	RETAIL		
ELICA	2	CUST SUPR	RETAIL	STANLEY INC.	TX
DEV		ADMIN	RETAIL		
NEWTON		DATA	RETAIL		
SAMUEL		IT	BANKING		
ELICA	2			STANLEY INC.	TX

LENGTH

The LENGTH function returns the length of a value.



The argument must be an expression that returns a value of any built-in data type that is not XML.

The result of the function is a large integer.

The result is the length of the argument. The length of a varying-length string is the actual length, not the maximum length.

The length of a graphic string is the number of double-byte characters. Unicode UTF-16 data is treated as graphic data; a UTF-16 supplementary character takes two DBCS characters to represent and as such is counted as two DBCS characters.

The length of all other values is the number of bytes used to represent the value:

- 2 for small integer
- 4 for large integer
- 8 for big integer

- The integer part of (p/2)+1 for decimal numbers with precision p
- 16 for DECFLOAT(34)
- 8 for DECFLOAT(16)
- 4 for single precision floating-point
- 8 for double precision floating-point
- The length of the string for strings
- 4 for DATE
- 3 for TIME
- 10 for TIMESTAMP
- 12 for TIMESTAMP WITH TIME ZONE
- 7+((p+1)/2) for TIMESTAMP(p)
- 9+((p+1)/2) for TIMESTAMP(p) WITH TIME ZONE
- The length of the row ID

Example 1:

The following example assigns the length of the string available in the column *EMPL_NAME* from the table EMPLOYEE to the variable *NAME_LENGTH*:

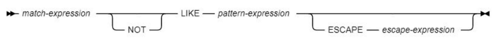
```
SELECT LENGTH(EMPL_NAME) AS NAME_LENGTH FROM EMPLOYEE
```

The above example returns the following.

```
NAME_LENGTH
6
6
5
3
6
6
5
```

LIKE

The LIKE predicate searches for strings that have a certain pattern.



The match-expression is the string to be tested for conformity to the pattern specified in pattern-expression. Underscore and percent sign characters in the pattern have a special meaning instead of their literal meanings unless escape-expression is specified.

The following example returns the rows where the *INDUSTRY* column value contains the string*RET* from the table EMPLOYEE.

```
SELECT * FROM EMPLOYEE WHERE INDUSTRY LIKE 'RET%'
```

The above example returns the following:

Table 84. Like

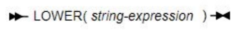
EMPL_NAME	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	RETAIL	50000	11/25/2019
CEASER	43	RETAIL	20000	

Table 84. Like (continued)

EMPL_NAME	DEPT	INDUSTRY	SALARY	DATEOFJOIN
ELICA	23	RETAIL	5000	10/10/2019
DEV	33	RETAIL	25000	11/27/2019
NEWTON	53	RETAIL	75000	11/25/2019

LOWER

The LOWER function returns a string in which all the characters are converted to lowercase characters.



string-expression

An expression that specifies the string to be converted. string-expression must return a value that is a built-in character or graphic string. A character string argument must not be a CLOB, and a graphic string argument must not be a DBCLOB. If string-expression is an EBCDIC graphic string, a blank string must not be specified for locale-name-string. If string-expression is bit data, locale-name-string must not be specified.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

Syntax alternatives:

LCASE is a synonym for LOWER. LOWER should be used for conformance to the SQL standard.

Example

The following example returns the characters in the variable *EMPL_NAME* in lowercase from the table EMPLOYEE.

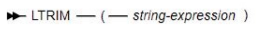
```
SELECT LOWER(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following:

```
bumrah
ceasar
elica
dev
newton
samuel
elica
```

LTRIM

The LTRIM function removes spaces from the beginning of a string expression.



The LTRIM function removes all of the spaces that are contained in the left side of the given string-expression.

string-expression
: An expression that specifies the source string. The argument must be an expression that returns a value that is a built-in string data type that is not a LOB, or a numeric data type. If the value is not a string data type, it is implicitly cast to VARCHAR before the function is evaluated. If string-expression is not FOR BIT DATA, trim-expression must not be FOR BIT DATA.

The result of the function depends on the data type of string-expression:

- VARCHAR if string-expression is a character string. If string-expression is defined as FOR BIT DATA the result is FOR BIT DATA.
- VARGRAPHIC if string-expression is a graphic string.
- VARBINARY if string-expression is a binary string.

The length attribute of the result is the same as the length attribute of string-expression.

Example:

The following example removes the spaces from the left side of the values in the *EMPL_NAME* from the table EMPLOYEE:

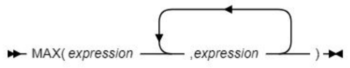
```
SELECT LTRIM(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH
CEASAR
ELICA
DEV
NEWTON
SAMUEL
ELICA
```

MAX

The MAX scalar function returns the maximum value in a set of values.



All but the first argument can be parameter markers. There must be two or more arguments.

Each argument must be an expression that returns a value of any built-in data type other than a CLOB, DBCLOB, BLOB, ROWID, or XML.

Character string arguments and binary string arguments cannot have a length attribute greater than 32704, and graphic string arguments cannot have a length attribute greater than 16352.

The arguments are evaluated in the order in which they are specified. The result of the function is the maximum argument value.

The selected argument is converted, if necessary, to the attributes of the result. If the MAX function has more than two arguments, the rules are applied to the first two arguments to determine a candidate result type. The rules are then applied to that candidate result type and the third argument to determine another candidate result type. This process continues until all arguments are analyzed and the final result type and CCSID is determined.

Notes

Syntax alternatives:
GREATEST is a synonym for MAX.

Example 1:

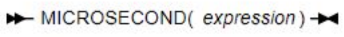
The following example selects the maximum value among the values in the *SALARY* column from the table EMPLOYEE.

```
SELECT MAX(SALARY) FROM EMPLOYEE
```

The above example returns *75000*.

MICROSECOND

The MICROSECOND function returns the microsecond part of a value.



The argument must be an expression that returns a value of one of the following built-in data types: a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a timestamp duration.

If the expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a timestamp or string representation of a timestamp**, the result is the microsecond part of the value, which is an integer between 0 and 999999. If the precision of the timestamp exceeds 6, the value is truncated.
- **If the argument is a duration**, the result is the microsecond part of the value, which is an integer between -999999 and 999999. A nonzero result has the same sign as the argument.

Example 1:

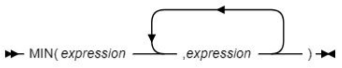
The following example returns the microsecond part of the string in the column *UPDATED_TS* in the table EMPL_COMP:

```
SELECT MICROSECOND(UPDATED_TS) FROM EMPL_COMP WHERE UPDATED_TS IS NOT NULL
```

The above example returns *454707*.

MIN

The MIN scalar function returns the minimum value in a set of values.



All but the first argument can be parameter markers. There must be two or more arguments.

Each argument must be an expression that returns a value of any built-in data type other than a CLOB, DBCLOB, BLOB, ROWID, or XML.

Character string arguments and binary string arguments cannot have a length attribute greater than 32704, and graphic string arguments cannot have a length attribute greater than 16352.

The arguments are evaluated in the order in which they are specified. The result of the function is the minimum argument value.

The selected argument is converted, if necessary, to the attributes of the result. If the MIN function has more than two arguments, the rules are applied to the first two arguments to determine a candidate result type. The rules are then applied to that candidate result type and the third argument to determine another candidate result type. This process continues until all arguments are analyzed and the final result type and CCSID is determined.

Notes

Syntax alternatives:
LEAST is a synonym for MIN.

Example 1:

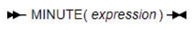
The following example selects the minimum value among the values in the *SALARY* column from the table EMPLOYEE

```
SELECT MIN(SALARY) FROM EMPLOYEE
```

The above example returns *20000*.

MINUTE

The MINUTE function returns the minute part of a value.



The argument must be an expression that returns a value of one of the following built-in data types: a time, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a time or timestamp duration.

If expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a time, timestamp, or string representation of either**, the result is the minute part of the value, which is an integer between 0 and 59.
- **If the argument is a time duration or timestamp duration**, the result is the minute part of the value, which is an integer between -99 and 99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example 1:

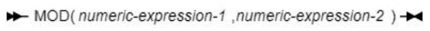
The following example selects the minute part of the string available in the *UPDATED_TS* column in the table EMPL_COMP.

```
SELECT MINUTE(UPDATED_TS) FROM EMPL_COMP WHERE UPDATED_TS IS NOT NULL
```

The above example returns *39*.

MOD

The MOD function divides the first argument by the second argument and returns the remainder.



The formula used to calculate the remainder is:

```
MOD(x,y) = x - FLOOR(x/y) * y
```

Where `x/y` is the truncated integer result of the division. The result is negative only if the first argument is negative.

Each argument must be an expression that returns a value of any built-in numeric data type.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The attributes of the result are based on the arguments as follows:

- If both arguments are large or small integers, the data type of the result is large integer.

- If both arguments are integers and at least one argument is a big integer, the data type of the result is big integer.
- If one argument is an integer and the other is a decimal, the data type of the result is decimal with the same precision and scale as the decimal argument.
- If both arguments are decimal, the data type of the result is decimal. The precision of the result is `min(p-s,p'-s') + max(s,s')` and the scale of the result is `max(s,s')`,
  - where the symbols p and s denote the precision and scale of the first argument, and the symbols p' and s' denote the precision and scale of the second argument.
  - If one argument is a floating-point number, and the other is not a DECFLOAT, or both argument is a floating-point number, the data type of the result is double precision floating-point.

    The operation is performed in floating-point. If necessary, the operands are first converted to double precision floating-point numbers. For example, an operation that involves a floating-point number and either an integer or a decimal number is performed with a temporary copy of the integer or decimal number that has been converted to double precision floating-point. The result of a floating-point operation must be within the range of floating-point numbers.
  - If either argument is a DECFLOAT, the data type of the result is DECFLOAT(34).

    If either argument is a special decimal floating point value, the general rules for arithmetic operations apply.

    If one argument is a DECFLOAT and the second argument is zero, the result is NaN and an invalid operation condition is returned.

Example:

The following example returns the remainder of dividing the value available in the *SALARY* column from the table EMPLOYEE by *3*.

```
SELECT MOD(SALARY, 3) FROM EMPLOYEE
```

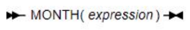
The above example returns the following.

```
2
2

1
0
0
```

MONTH

The MONTH function returns the month part of a value.



The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a date or timestamp duration.

If expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument:

- **If the argument is a date, timestamp, or string representation of either**, the result is the month part of the value, which is an integer between 1 and 12.
- **If the argument is a date duration or timestamp duration**, the result is the month part of the value, which is an integer between -99 and 99. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example 1:

The following example selects the month part in the value available in the *UPDATED_TS* column in the table EMPLOYEE:

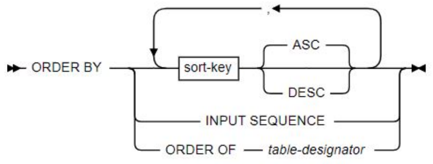
```
SELECT MONTH(UPDATED_TS) FROM EMPL_COMP WHERE UPDATED_TS IS NOT NULL
```

The above example returns *11*

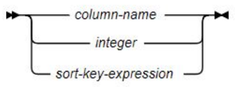
ORDER BY

The ORDER BY clause specifies an ordering of the rows of the result table.

order-by-clause



sort-key:



A subselect that contains an ORDER BY clause cannot be specified in the outermost fullselect of a view.

If the subselect is not enclosed within parentheses and is not the outermost fullselect, the ORDER BY clause cannot be specified. The ORDER BY clause cannot be used in an outermost fullselect that contains a FOR UPDATE clause.

An ORDER BY clause that is specified in a subselect only affects the order of the rows that are returned by the query if the subselect is the outermost fullselect, except when a nested subselect includes an ORDER BY clause and the outermost fullselect specifies that the ordering of the rows should be retained (by using the ORDER OF table-designator clause).

Multiple ORDER BY clauses can be specified in the same subselect if each clause is separated with parentheses.

INPUT SEQUENCE
Indicates that the result table reflects the input order of the rows specified in the VALUES clause of an INSERT statement. INPUT SEQUENCE ordering can be specified only when an INSERT statement is specified in a from-clause.

ORDER OF table-designator
Specifies that the same ordering of the rows for the result table that is designated by table-designator should be applied to the result table of the subselect (or fullselect) that contains the ORDER OF specification. There must be a table reference in the FROM clause of the subselect (or fullselect) that specifies this clause and matches table-designator.

For an ORDER BY clause in an OLAP specification, table-designator must not specify a table function, a collection-derived table, a materialized view, a nested table expression that is materialized, an alias, or a synonym.

sort-key

A column-name, integer, or sort-key-expression that specifies the value that is to be used to order the rows of the result of the subselect.

If a single sort-key is identified, the rows are ordered by the values of that sort-key. If more than one sort-key is identified, the rows are ordered by the values of the first sort-key, then by the values of the second sort-key, and so on. A sort-key cannot be a LOB or XML expression.

The result table can be ordered by a named column in the select list by specifying a sort-key that is an integer or the column name. The result table can be ordered by an unnamed column in the select list by specifying a sort-key that is an integer or, in some cases, by a sort-key-expression that matches the expression in the select list.

column-name

An identifier that usually identifies a column of the result table. In this case, column-name must be the name of a named column in the select list. If the fullselect includes a set operator, the column name cannot be qualified.

If the query is a subselect, the column-name can also identify a column name of a table, view, or nested table expression identified in the FROM clause, including a column that is defined as implicitly hidden. The subselect must not include any of the following:

- DISTINCT in the select list
- Aggregate functions in the select list
- GROUP BY clause

integer

An unsigned integer that must be greater than 0 and not greater than the number of columns in the result table. The integer n identifies the nth column of the result table.

sort-key-expression

An expression that is not simply a column-name or unsigned integer constant. The query to which ordering is applied must be a subselect to use this form of the sort-key.

The sort-key-expression cannot include an expression that is not deterministic or a function that is defined to have an external action except for the RID built-in function and the ROW CHANGE expression. If sort-key-expression includes an aggregate function, the input arguments to that function must not reference a named column in the select list that is derived from an aggregate function.

If DISTINCT is used in the select list of the subselect, sort-key-expression must match an expression in the select list of the subselect. Scalar-fullselects are never matched.

If the subselect is grouped, the sort-key-expression might or might not be in the select list of the subselect. When sort-key-expression is not in the select list the following rules apply:

- Each expression in the ORDER BY clause must either:
  - Use one or more grouping expressions
  - Use a column name that either unambiguously identifies a grouping column of R or is specified within a aggregate function.
- Each expression in the ORDER BY clause must not contain a scalar fullselect.

ASC

Uses the values of the sort-key in ascending order.

ASC is the default.

DESC

Uses the values of the sort-key in descending order.

The null value is higher than all other values. If your ordering specification does not determine a complete ordering, rows with duplicate values of the last identified sort-key have an arbitrary order. If you do not specify ORDER BY, the rows of the result table have an arbitrary order.

Column access controls do not effect the operation of the ORDER BY clause. The order is based on the original column values. However, after column masks are applied, the masked values in the final result table might not reflect the order of the original column values.

Column names in sort keys: A column name in a sort-key must conform to the following rules:

- If the column name is qualified, the query must be a subselect. The column name must unambiguously identify a column of a table, view, or nested table expression in the FROM clause of the subselect; its value is used to compute the value of the sort specification.
- If the column name is unqualified and the query is a subselect:
  - If the column name is identical to the name of more than one column of the result table, the column name must unambiguously identify a column of some table, view, or nested table expression in the FROM clause of the ordering subselect.
  - If the column name is identical is one column of the result table, its value is used to compute the value of the sort specification.
  - If the column name is not identical to a column in the result table, it must unambiguously identify a column of a table, view, or nested table expression in the FROM clause of the subselect. If the column name is identical to one column of a table, view, or nested table expression in the FROM clause of the subselect, its value is used to compute the value of the sort specification.

Example:

The following example sorts the rows from the table EMPLOYEE in an ascending order.

```
SELECT EMPL_NAME, AGE FROM EMPLOYEE ORDER BY AGE ASC
```

The above example returns the following:

```
EMPL_NAME , AGE
ELICA , 23
DEV , 33
CEASAR , 43
SAMUEL , 43
BUMRAH , 53
NEWTON , 53
ELICA ,
```

OUTER JOIN

An outer join is a method of combining two or more tables so that the result includes unmatched rows of one of the tables, or of both tables. The matching is based on the join condition.

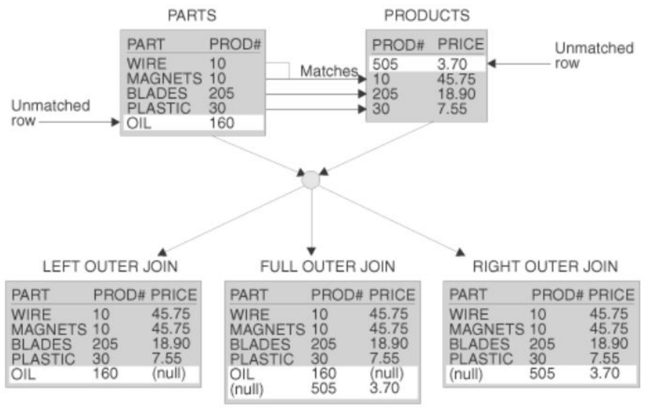
DVM supports three types of outer joins:

full outer join
Includes unmatched rows from both tables. If any column of the result table does not have a value, that column has the null value in the result table.

left outer join
Includes rows from the table that is specified before LEFT OUTER JOIN that have no matching values in the table that is specified after LEFT OUTER JOIN.

right outer join
Includes rows from the table that is specified after RIGHT OUTER JOIN that have no matching values in the table that is specified before RIGHT OUTER JOIN.

The following table illustrates how the sample PARTS and PRODUCTS tables can be combined using the three outer join functions.

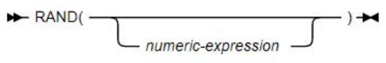


The result table contains data that is joined from all of the tables, for rows that satisfy the search conditions.

The result columns of a join have names if the outermost SELECT list refers to base columns. However, if you use a function (such as COALESCE or VALUE) to build a column of the result, that column does not have a name unless you use the AS clause in the SELECT list.

RAND

The RAND function returns a random floating-point value between 0 and 1. An argument can be specified as an optional seed value.



numeric-expression
If numeric-expression is specified, it is used as the seed value. The argument must be an expression that returns a value of a built-in integer data type (SMALLINT or INTEGER). The value must be between 0 and 2,147,483,646.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) and then assigned to an INTEGER value.

The result of the function is a double precision floating-point number.

A specific seed value, other than zero, will produce the same sequence of random numbers for a specific instance of a RAND function in a query each time the query is executed. The seed value is used only for the first invocation of an instance of the RAND function within a statement. `RAND(0)` is processed the same as `RAND()`.

The RAND function is a non-deterministic function.

Example:

The following example assigns a random number for the column *EMPL_ID* in the table EMPLOYEE.

```
INSERT INTO EMPL_COMP (EMPL_ID, EMPL_NAME) VALUES(RANDOM(),'KANE');
```

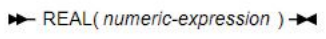
The above example assigns a random value for the *EMPL_ID* column for the employee *KANE* as shown below.

```
EMPL_ID, EMPL_NAME, COMPANY, LOCATION, UPDATED_TS
1 , AKHIL , TIMBER INC. , NY , 2019-11-26 05:39:59.454707
2 , ELICA , STANLEY INC. , TX ,
3 , SMITH , CHROME INC. , WA ,
1919574970 , KANE ,  ,  ,
```

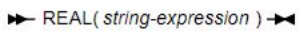
REAL

The REAL function returns a single-precision floating-point representation of either a number or a string representation of a number.

Numeric to Real:



String to Real:



Numeric to Real

numeric-expression
An expression that returns a value of any built-in numeric data type.

The result is the same number that would occur if the argument were assigned to a single precision floating-point column or variable. If the numeric value of the argument is not within the range of single precision floating-point, an error occurs.

String to Real

string-expression
An expression that returns a value of a character or graphic string (except a CLOB or DBCLOB) with a length attribute that is not greater than 255 bytes. The string must contain a valid string representation of a number.

The result is the same number that would result from CAST(`string-expression AS REAL`). Leading and trailing blanks are eliminated and the resulting string must conform to the rules for forming an SQL floating-point, integer, or decimal constant.

The result of the function is a single precision floating-point number.

Notes

Syntax alternatives:
To increase the portability of applications, use the CAST specification.

Examples

Example 1:
Using the sample table EMPLOYEE, find the ratio of salary to age for employees. The columns involved, SALARY and AGE, have decimal data types. To express the result in single precision floating-point, apply REAL to SALARY so that the division is carried out in floating-point (actually double precision) and then apply REAL to the complete expression so that the results are returned in single precision floating-point.

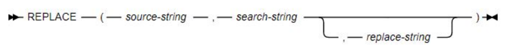
```
SELECT EMPL_NAME, REAL(SALARY/AGE) FROM EMPLOYEE
    WHERE AGE > 0;
```

The above example returns the following.

```
EMPL_NAME,
BUMRAH , 943.396240234375
CEASAR , 465.1162109375
ELICA ,
DEV , 757.57568359375
NEWTON , 1415.09423828125
SAMUEL , 1744.18603515625
```

REPLACE

The REPLACE function replaces all occurrences of a search-string in a source-string with a replace-string. If the search-string is not found in the source-string, the source-string is returned unchanged.



source-string
An expression that specifies the source string. The expression must return a value that is a built-in character string, graphic string, or binary string data type that is not a LOB and it cannot be an empty string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

search-string
An expression that specifies the string to be removed from the source string. The expression must return a value that is a built-in character string, graphic string, or binary string data type that is not a LOB; the value cannot be an empty string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

replace-string
An expression that specifies the replacement string. The expression must return a value that is a built-in character string, graphic string, or binary string data type that is not a LOB.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

If replace-string is not specified or is an empty string, nothing replaces the string that is removed from the source string.

The actual length of each string must be 32764 bytes or less for character and binary strings or 16382 or less for graphic strings.

All three arguments must have compatible data types. If the expressions have different CCSID sets, then the expressions are converted to the CCSID set of source-string.

The data type of the result of the function depends on the data type of source-string, search-string, and replace-string:

- VARCHAR if source-string is a character string. The encoding scheme of the result is the same as source-string. The CCSID of the result depends on the arguments:
  – If source-string, search-string, or replace-string is bit data, the result is bit data.
  – If source-string, search-string, and replace-string are all SBCS Unicode data, the CCSID of the result is the CCSID for SBCS Unicode data.
  – If source-string is SBCS Unicode data, and search-string or replace-string is not SBCS Unicode data, the CCSID of the result is the mixed CCSID for Unicode data.
  – Otherwise, the CCSID of the result is the mixed CCSID that corresponds to the CCSID of source-string. However, if the input is EBCDIC or ASCII and there is no corresponding system CCSID for mixed, the CCSID of the result is the CCSID of source-string.
- VARGRAPHIC if source-string is a graphic. The encoding scheme of the result is the same as source-string. The CCSID of the result is the same as the CCSID of source-string.
- VARBINARY if source-string, search-string, and replace-string are binary strings.

The length attribute of the result depends on the arguments:

- If the length attribute of replace-string is less than or equal to the length attribute of search-string, the length attribute of the result is the length attribute of source-string.

- If the length attribute of replace-string is greater than the length attribute of search-string, the length attribute of the result is determined as follows depending on the data type of the result:
  - For VARCHAR or VARBINARY:
    - If `L1 < = 4000`, the length attribute of the result is `MIN(4000, (L3*(L1/L2)) + MOD(L1,L2))`
    - Otherwise, the length attribute of the result is `MIN( 32764 , (L3*(L1/L2)) + MOD(L1,L2))`
  - For VARGRAPHIC:
    - If `L1 < = 2000`, the length attribute of the result is `MIN(2000, (L3*(L1/L2)) + MOD(L1,L2))`
    - Otherwise, the length attribute of the result is `MIN( 16382 , (L3*(L1/L2)) + MOD(L1,L2))`

  where:
  - L1 is the length attribute of source-string
  - L2 is the length attribute of search-string if the search string is a string constant. Otherwise, L2 is 1.
  - L3 is the length attribute of replace-string

If the result is a character string or binary string, the length attribute of the result must not exceed 32764 . If the result is a graphic string, the length attribute of the result must not exceed 16382 .

The actual length of the result is the actual length of source-string plus the number of occurrences of search-string that exist in source-string multiplied by the actual length of replace-string minus the actual length of search-string. If the actual length of the result string exceeds the maximum for the return data type, an error occurs.

Example:

The following example replace all occurrences of *CEASAR* with *GEORGE* in the string available in the column *EMPL_NAME* from the table EMPLOYEE.

```
SELECT REPLACE(EMPL_NAME,'CEASAR','GEORGE') FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH
GEORGE
ELICA
DEV
NEWTON
SAMUEL
ELICA
```

RIGHT OUTER JOIN

The RIGHT OUTER JOIN clause lists rows from the right table even if there are no matching rows on left table.

As in an inner join, the join condition of a right outer join can be any simple or compound search condition that does not contain a subquery reference

Example

The following example joins the tables Table 72 on page 327 and Table 73 on page 328.

```
SELECT A.ID, A.NAME, A.DEPT, A.JOB, A.YEARS, A.SALARY, A.COMM
FROM JOIN1;
RIGHT OUTER JOIN
JOIN2 B;
ON A.ID=B.ID
```

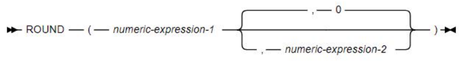
The query returns the following.

Table 85. RIGHT OUTER JOIN

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
999	JOHN		SALES		16.99	
10	MIC	20	MGR	7	98357	0.00
20	WILLIAM	20	SALES	8	78171	612.45
30	MIC	38	MGR	5	77506	0.00
40	OBRIEN	38	SALES	6	78006	846.55

ROUND

The ROUND function returns a number that is rounded to the specified number of places to the right or left of the decimal place.



numeric-expression-1
An expression that returns a value of any built-in numeric data type.

If expression-1 is a decimal floating-point data type, the DECFLOAT ROUNDING MODE will not be used. The rounding behavior of ROUND corresponds to a value of ROUND_HALF_UP. If you want a different rounding behavior, use the QUANTIZE function.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

numeric-expression-2

An expression that returns a value that is a built-in numeric, character string or graphic string data type. If the value is not of type INTEGER, it is implicitly cast to INTEGER before evaluating the function.

The absolute value of integer specifies the number of places to the right of the decimal point for the result if numeric-expression-2 is not negative. If numeric-expression-2 is negative, numeric-expression-1 is rounded to the sum of the absolute value of numeric-expression-2+1 number of places to the left of the decimal point.

If the absolute value of numeric-expression-2 is larger than the number of digits to the left of the decimal point, the result is 0. (For example, `ROUND(748.58,-4)` returns 0.)

If numeric-expression-1 is positive, a digit value of 5 is rounded to the next higher positive number. If numeric-expression-1 is negative, a digit value of 5 is rounded to the next lower negative number.

The result of the function has the same data type and length attribute as the first argument except that the precision is increased by one if the argument is DECIMAL and the precision is less than 31. For example, an argument with a data type of DECIMAL(5,2) results in DECIMAL(6,2). An argument with a data type of DECIMAL(31,2) results in DECIMAL(31,2).

Example 1:

The following example rounds the number available in the *SALARY*column in the table EMPLOYEE.

```
SELECT ROUND(SALARY) FROM EMPLOYEE
```

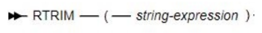
The above example returns the following.

```
50000
20000

25000
75000
75000
```

RTRIM

The RTRIM function removes spaces from the right side of a string expression.



The RTRIM function removes all of the spaces contained at the right side of a string-expression.

string-expression
: An expression that specifies the source string. The argument must be an expression that returns a value that is a built-in string data type that is not a LOB, or a numeric data type. If the value is not a string data type, it is implicitly cast to VARCHAR before the function is evaluated. If string-expression is not FOR BIT DATA, trim-expression must not be FOR BIT DATA.

The result of the function depends on the data type of string-expression.

- VARCHAR if string-expression is a character string. If string-expression is defined as FOR BIT DATA, the result is FOR BIT DATA.
- VARGRAPHIC if string-expression is a graphic string.
- VARBINARY if string-expression is a binary string.

The length attribute of the result is the same as the length attribute of string-expression.

Example:
The following example removes spaces at the right side of the values in the *EMPL_NAME* from the table EMPLOYEE:

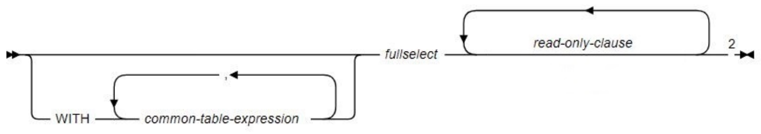
```
SELECT RTRIM(EMPL_NAME) FROM EMPLOYEE
```

The above example returns the following.

```
BUMRAH
CEASAR
ELICA
DEV
NEWTON
SAMUEL
ELICA
```

SELECT

The SELECT statement made up a series of clauses that are defined by SQL as being executed in a logical order. SELECT statements allow users to definite and organize information that is retrieved from a specified table.



Note:

- The read-only-clause must not be specified if an update-clause is specified.
- The same clause must not be specified more than one time.

The tables and the view identified in a select statement can be at the current server or any subsystem with which the current server can establish a connection.

For local queries or remote queries, if a table is encoded as ASCII or Unicode, the retrieved data is encoded in EBCDIC.

A select statement can implicitly or explicitly invoke user-defined functions or implicitly invoke stored procedures. This technique is known as nesting of SQL statements. A function or procedure is implicitly invoked in a select statement when it is invoked at a lower level. For instance, if you invoke a user-defined function from a select statement and the user-defined function invokes a stored procedure, you are implicitly invoking the stored procedure.

- **common-table-expression**

  A common table expression defines a result table with table-identifier that can be referenced in any FROM clause of the fullselect that follows.

- **read-only-clause**

  The read-only clause specifies that the result table is read-only. Therefore, the cursor cannot be referred to in positioned UPDATE or DELETE statements.

Example

SELECT - All Rows

The following example selects all rows from the table EMPLOYEE.

```
SELECT * FROM EMPLOYEE;
```

Result

Table 86. SELECT ALL

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					

SELECT - Limited Columns and Row

The following example selects only the columns EMPL_NAME, AGE, and limits the records to 3.

```
SELECT EMPL_NAME, AGE
FROM EMPLOYEE LIMIT 3;
```

Result

Table 87. SELECT - Limited Columns and Row

EMPL_NAME	AGE
BUMRAH	53
CEASAR	43
ELICA	23

SELECT - Condition

The following example applies the condition of selecting only the record where the EMPL_NAME has LOCATION = 'TX' in the table EMPL_COMP.

```
SELECT EMPL_NAME, AGE, DEPT, INDUSTRY, SALARY, DATEOFJOIN
FROM EMPLOYEE
WHERE EMPL_NAME = (SELECT EMPL_NAME FROM EMPL_COMP WHERE LOCATION = 'TX');
```

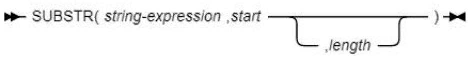
Result

Table 88. SELECT - Condition

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
ELICA					

SUBSTR

The SUBSTR function returns a substring of a string.



string-expression
An expression that specifies the string from which the result is derived. The string must be a character, graphic, or binary string. If string-expression is a character string, the result of the function is a character string. If it is a graphic string, the result of the function is a graphic string. If it is a binary string, the result of the function is a binary string.

The argument can also be a numeric data type. The numeric argument is implicitly cast to a VARCHAR data type.

A substring of string-expression is zero or more contiguous characters of string-expression. If string-expression is a graphic string, a character is a DBCS character. If string-expression is a character string or a binary string, a character is a byte. The SUBSTR function accepts mixed data strings. However, because SUBSTR operates on a strict byte-count basis, the result will not necessarily be a properly formed mixed data string.

start
An expression that specifies the position within string-expression to be the first character of the result. The value of the large integer must be between 1 and the length attribute of string-expression. (The length attribute of a varying-length string is its maximum length.) A value of 1 indicates that the first character of the substring is the first character of string-expression.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) which is then assigned to an INTEGER value.

length

An expression that specifies the length of the resulting substring. If specified, length must be an expression that returns a value that is a built-in large integer data type.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34) which is then assigned to an INTEGER value.

The value must be greater than or equal to 0 and less than or equal to n, where n is the length attribute of `string-expression - start + 1`. The specified length must not, however, be the large integer constant 0.

If length is explicitly specified, string-expression is effectively padded on the right with the necessary number of characters so that the specified substring of string-expression always exists. Hexadecimal zeros are used as the padding character when string-expression is binary data. Otherwise, a blank is used as the padding character.

If string-expression is a fixed-length string, omission of length is an implicit specification of `LENGTH(string-expression) - start + 1`, which is the number of characters (or bytes) from the character (or byte) specified by start to the last character (or byte) of string-expression. If string-expression is a varying-length string, omission of length is an implicit specification of the greater of zero or `LENGTH(string-expression) - start + 1`. If the resulting length is zero, the result is an empty string.

If length is explicitly specified by a large integer constant that is 255 or less, and string-expression is not a LOB, the result is a fixed-length string with a length attribute of length. If length is not explicitly specified, but string-expression is a fixed-length string and start is an integer constant, the result is a fixed-length string with a length attribute equal to `LENGTH(string-expression) - start + 1`. In all other cases, the result is a varying-length string. If length is explicitly specified by a large integer constant, the length attribute of the result is length; otherwise, the length attribute of the result is the same as the length attribute of string-expression.

The CCSID of the result is the CCSID of string-expression.

Example 1:

The following example returns the substring from the value in the *INDUSTRY*column from the table EMPLOYEE.

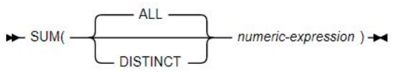
```
SELECT SUBSTR(INDUSTRY,1,3) FROM EMPLOYEE
```

The above example returns the following.

```
RET
RET
RET
RET
RET
BAN
```

SUM

The SUM function returns the sum of a set of numbers.



The argument values can be of any built-in numeric data type, and their sum must be within the range of the data type of the result.

The arguments can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

The data type of the result is determined as follows:

- DECFLOAT(34) if the argument is DECFLOAT(n).
- Large integer if the argument is small integer.
- Double precision floating-point if the argument is single precision floating-point.
- Otherwise, the result is the same as the data type of the argument.

If the data type of the argument values is decimal, the scale of the result is the same as the scale of the argument values, and the precision of the result depends on the precision of the argument values and the decimal precision option:

- If the precision of the argument values is greater than 15 or the DEC31 option is in effect, the precision of the result is `min(31,P+10)`, where P is the precision of the argument values.
- Otherwise, the precision of the result is 15.

Example:

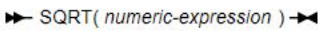
The following example selects the sum of the values in the numerical column *SALARY* from the table EMPLOYEE.

```
SELECT SUM(SALARY) FROM EMPLOYEE
```

The above examples returns *250000*

SQRT

The SQRT function returns the square root of the argument.



The argument must be an expression that returns the value of any built-in numeric data type. If the argument is DECFLOAT, the operation is performed in DECFLOAT. Otherwise, the argument is converted to a double precision floating-point number for processing by the functions.

The argument can also be a character string or graphic string data type. The string input is implicitly cast to a numeric value of DECFLOAT(34).

If the argument is DECFLOAT(n), the result is DECFLOAT(n). Otherwise, the result of the function is a double precision floating-point number. If the argument is a special decimal floating point value, the general rules for arithmetic operations apply.

Example:

The following example returns the square root value of the numerical column *SALARY* from the table EMPLOYEE.

```
SELECT SQRT(SALARY) FROM EMPLOYEE
```

The above example returns the following:

```
223.60679774997897
141.42135623730948
70.71067811865474
158.11388300841895
273.8612787525831
273.8612787525831
```

UNION

Using the UNION keyword, you can combine two or more subselects to form a fullselect.

When SQL encounters the UNION keyword, it processes each subselect to form an interim result table, it combines the interim result table of each subselect and deletes duplicate rows to form a combined result table. You can use different clauses and techniques when coding select-statements.

The combined list is derived from two tables and contains no duplicates.

To better understand the results from these SQL statements, imagine that SQL goes through the following process:

When you use UNION:

- Any ORDER BY clause must appear after the last subselect that is part of the union. The ORDER BY clause specifies that the combined result table is to be in collated sequence. ORDER BY is not allowed in a view.
- A name may be specified on the ORDER BY clause if the result columns are named. A result column is named if the corresponding columns in each of the unioned select-statements have the same name. An AS clause can be used to assign a name to columns in the select list.

To identify which subselect each row is from, you can include a constant at the end of the select list of each subselect in the union. When SQL returns your results, the last column contains the constant for the subselect that is the source of that row. For example, you can specify:

```
SELECT A, B, 'A1' ...
UNION
SELECT X, Y, 'B2'...
```

When a row is returned, it includes a value (either A1 or B2) to indicate the table that is the source of the row's values. If the column names in the union are different, SQL uses the set of column names specified in the first subselect when interactive SQL displays or prints the results, or in the SQLDA resulting from processing an SQL DESCRIBE statement.

Note: Sort sequence is applied after the fields across the UNION pieces are made compatible. The sort sequence is used for the distinct processing that implicitly occurs during UNION processing.

The following example combines the records from the tables Table 70 on page 327 and Table 71 on page 327.

```
SELECT A.EMPL_NAME, A.AGE, A.DEPT, A.INDUSTRY, A.SALARY, A.DATEOFJOIN
FROM EMPLOYEE A
UNION
SELECT B.EMPL_NAME, B.AGE, B.DEPT, B.INDUSTRY, B.SALARY, B.DATEOFJOIN
FROM EMP B;
```

The query combines all the records and leaves out the duplicate records.

Table 89. VIRTUAL TABLE - EMPLOYEE

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					

Table 89. VIRTUAL TABLE - EMPLOYEE (continued)

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
WAYNE	36	ADMIN	RETAIL	45000	2018-11-01

UNION ALL

If you want to keep duplicates in the result of a UNION operation, specify the UNION ALL keyword instead of just UNION.

This topic uses the same steps and example as "UNION" on page 365.

The following example combines all the records from the tables Table 70 on page 327 and Table 71 on page 327.

```
SELECT A.EMPL_NAME, A.AGE, A.DEPT, A.INDUSTRY, A.SALARY, A.DATEOFJOIN
FROM EMPLOYEE A
UNION ALL
SELECT B.EMPL_NAME, B.AGE, B.DEPT, B.INDUSTRY, B.SALARY, B.DATEOFJOIN
FROM EMP B;
```

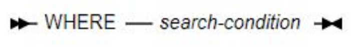
The query includes all the records including the duplicate records.

Table 90. VIRTUAL TABLE - EMPLOYEE

EMPL_NAME	AGE	DEPT	INDUSTRY	SALARY	DATEOFJOIN
BUMRAH	53	SALES	RETAIL	50000	2019-11-25
CEASAR	43	MARKETING	RETAIL	20000	
ELICA	23	CUST SUPR	RETAIL	5000	2019-10-10
DEV	33	ADMIN	RETAIL	25000	2019-11-27
NEWTON	53	DATA	RETAIL	75000	2019-11-25
SAMUEL	43	IT	BANKING	75000	2019-11-25
ELICA					
DEV	33	ADMIN	RETAIL	25000	2019-11-27
WAYNE	36	ADMIN	RETAIL	45000	2018-11-01

WHERE

The WHERE clause specifies a result table that consists of those rows of R for which the search condition is true. R is the result of the FROM clause of the subselect.



The search condition must conform to the following rules:

- Each column name must unambiguously identify a column of R or be a correlated reference. A column name is a correlated reference if it identifies a column of a table, view, common-table-expression, or nested-table-expression that is identified in an outer subselect.
- An aggregate function must not be specified unless the WHERE clause is specified in a subquery of a HAVING clause and the argument of the function is a correlated reference to a group.

Any subquery in the search-condition is effectively executed for each row of R and the results are used in the application of the search-condition to the given row of R. A subquery is actually executed for each row of R only if it includes a correlated reference. In fact, a subquery with no correlated references is executed just one time, whereas a subquery with a correlated reference might have to be executed one time for each row.

The column access control does not affect the operation of the WHERE clause.

Example:

The following example returns the rows from the table EMPLOYEE with columns *EMPL_NAME* and *AGE* where *AGE* is greater than *50*.

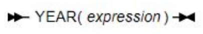
```
SELECT EMPL_NAME,AGE FROM EMPLOYEE
WHERE AGE >50
```

The above example returns the following:

```
BUMRAH , 53
NEWTON , 53
```

YEAR

The YEAR function returns the year part of a value that is a character or graphic string. The value must be a valid string representation of a date or timestamp.



The argument must be an expression that returns one of the following built-in data types: a date, a timestamp, a character string, a graphic string, or a numeric data type.

- If expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or timestamp with an actual length of not greater than 255 bytes.
- If expression is a number, it must be a date or timestamp duration.

If the expression is a timestamp with a time zone, or a valid string representation of a timestamp with a time zone, the result is determined from the UTC representation of the datetime value.

The result of the function is a large integer.

The other rules depend on the data type of the argument specified:

- **If the argument is a date, a timestamp, or a string representation of either**, the result is the year part of the value, which is an integer between 1 and 9999.
- **If the argument is a date duration or a timestamp duration**, the result is the year part of the value, which is an integer between -9999 and 9999. A nonzero result has the same sign as the argument.
- **If the argument contains a time zone**, the result is the year part of the value expressed in UTC.

Example 1:

The following example selects the year part from the column *UPDATED_TS* from the table EMPL_COMP.

```
SELECT YEAR(UPDATED_TS) FROM EMPL_COMP
```

The above example returns *2019*.

Chapter 13. SQL DMF supported data types

This appendix contains the language-specific data definitions that are used by the Data Mapping Facility (DMF) and shows the equivalent SQL data types that are used by IBM Data Virtualization Manager for z/OS. It also shows the SQL data types that are supported by the different interfaces in IBM Data Virtualization Manager for z/OS.

Adabas

Although it is not a programming language, Adabas has a file definition, created by the Adabas database administrator, that is used to generate a map.

Table 91. Data definitions for Adabas

Data definition	SQL type	Host format
A - Alphanumeric	SQL_Char	Character
B - Binary	SQL_Binary	Binary
F - Fixed point	Length 2 SQL_Smallint	Smallint
	Length 4 SQL_Integer	Integer
	Length 8 SQL_BigInt	BigInt
G - Floating point	Length 4 SQL_Float	Float
	Length 8 SQL_Double	Float
P - Packed decimal	SQL_Decimal	Packed Decimal
	Length 4 SQL_Date	Date
	Length 7 SQL_Timestamp	Timestamp
U - Unpacked decimal	SQL_Char	Unpacked decimal
W – Wide Alphanumeric	Not supported	Not supported

COBOL

Table 92. Data definitions for COBOL

Data definition	SQL data type	Host format
PIC X(30) PIC A(30)	SQL_Char	Character
PIC S9(3)V9(3) PIC S9(3)V9(3) USAGE DISPLAY	SQL_Char	Display Numeric
PIC G(30) USAGE DISPLAY-1	SQL_Graphic (SQL_Unicode)	Graphic (DBCS)

Table 92. Data definitions for COBOL (continued)

Data definition	SQL data type	Host format
PIC S9(_) USAGE BINARY PIC S9(_) USAGE COMP PIC S9(_) USAGE COMP-4	Length 1 to 4 SQL_Smallint Length 5 to 9 SQL_Integer Length 10 to 18 SQL_Binary	Smallint Integer Binary **Note:** Fields with a length of 10 to 18 become SQL_BIGINT when support for BIGINT is added.
USAGE IS COMP-1	SQL_Float	Float
USAGE IS COMP-2	SQL_Double	Float
PIC S9(03)V9(3) USAGE COMP-3 PIC S9(03)V9(3) USAGE PACKED-DECIMAL	SQL_Decimal	Packed Decimal
PIC S9(_) USAGE COMP-5 PIC 9(_) USAGE COMP-5	Length 1 to 4 SQL_Smallint Length 5 to 9 SQL_Integer Length 10 to 18 SQL_Binary	Smallint Integer Binary **Note:** Fields with a length of 10 to 18 become SQL_BIGINT when support for BIGINT is added.

Table 93. PIC S9(_) USAGE COMP-5

Picture	Storage representation	Numeric values

S9(1) through S9(4)
S9(5) through S9(9)
S9(10) through S9(18)
	Binary half-word (2 bytes)
	Binary full-word (4 bytes)
	Binary double-word (8 bytes)
		-32768 to +32767
		-2,147,483,648 to +2,147,483,647
		-9,223,372,036,854,775,808 to +9.223,372,036,854,775,807

Table 94. PIC 9(_) USAGE COMP-5

Picture	Storage representation	Numeric values

9(1) through 9(4)
9(5) through 9(9)
9(10) through 9(18)
	Binary half-word (2 bytes)
	Binary full-word (4 bytes)
	Binary double-word (8 bytes)
		0 to 65535
		0 to 4,294,967,295
		0 to 18,446,744,073,709,551,615

IMS - DBD (database description)

A database description defines an IMS database. To increase the available data type, merge a COBOL map with the database description.

Table 95. Data definitions for IMS - DBD

Data definition	SQL type	Host format
TYPE=C - Alphanumeric	SQL_Char	Character
TYPE =X - Hexadecimal	SQL_Binary	Binary
TYPE=P - Packed Decimal	SQL_Decimal	Packed Decimal
TYPE=F - Binary Fullword **Note:** Only valid for MSDB databases.	SQL_Integer	Integer
TYPE=H - Binary Halfword **Note:** Only valid for MSDB databases.	SQL_Smallint	Smallint

Natural conversions

The table describes how Natural data types are converted to ODBC data types.

Natural	ODBC
A-Alphanumeric	SQL_CHAR
B-Binary	(If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
C-Attribute Control	N/A
D-Date	*SQL_DECIMAL
F-Floating Point	(If 4 bytes) SQL_FLOAT (If 8 bytes) SQL_DOUBLE
I-Integer	(If 1 byte) SQL_BINARY (If 2 bytes) SQL_SMALLINT (If 4 bytes) SQL_INTEGER
L-Logical	SQL_BINARY
N-Numeric	SQL_NUMERIC
P-Packed	SQL_DECIMAL
T-Time	*SQL_DECIMAL

Note: Although the IBM Data Virtualization Manager for z/OS Interface for ADABAS supports the conversion of ODBC date and time to the Natural date and time format, the IBM Data Virtualization Manager for z/OS Interface for Natural only allows the passing of the internal format for date and time (P6 and P12, respectively).

Natural DDM (data definition module)

A DDM is a file that is used to create a view of an ADABAS file. It is used to provide long column names and to limit the view to a subset of the fields that are defined in the Adabas file.

Table 96. Data definitions for Natural DDM

Data definition	SQL type	Host format
A – Alphanumeric	SQL_Char	Character
B - Binary	SQL_Binary	Binary
F - Fixed point	Length 2 SQL_Smallint	Smallint
	Length 4 SQL_Integer	Integer
G - Floating point	Length 4 SQL_Float	Float
	Length 8 SQL_Double	Float
P - Packed decimal	SQL_Decimal	Packed Decimal
N – Unpacked decimal	SQL_Char	Unpacked decimal
D - Date	SQL_Date	
T - Time	SQL_Time	Not supported
	SQL_Graphic	Graphic (DBCS)

SQL Type Support by the IBM Data Virtualization Manager for z/OS interface

SQL Type	VSAM/ CICS VSAM	Adabas	ACI	SQL/IMS	CICS SP	IMS SP
SQL_Char	X	X	X	X	X	X
SQL_Numeric	X	X	X	X	X	X
SQL_Decimal	X	X	X	X	X	X
SQL_Bigint						
SQL_Integer	X	X	X	X	X	X
SQL_Smallint	X	X	X	X	X	X
SQL_Float						
SQL_Real						
SQL_Double						
SQL_Date		X		X		
SQL_Time	X	X				
SQL_Binary	X	X	X	X	X	X
SQL_Graphic	X	X	X	X	X	X

Index

SC27-9303-00