Updates that apply to IBM® DB2® Analytics Accelerator Loader for z/OS® V2R1 User's Guide (SC27-6777-00)

Date of change: August 2017
Topic: Multiple
Change description: Documentation changes made in support of PTFs UI49612 and UI49614 APARs PI76778, PI80385, PI80386, PI80783, PI81983, and PI83170 – Enhancements for Accelerator Loader server

The following topics have been updated or introduced:

In chapter "Overview":
Topic “What’s new”

In chapter "Preparing to customize":
Topic: "Worksheets: Gathering parameter values for Tools Customizer"

In chapter "Customizing DB2 Analytics Accelerator Loader":
Topic: "IMS data access methods"
Topic: "Modifying the server configuration member for IMS Direct"
Topic: "Configuring access to System Management Facility (SMF) files"
Topic: "Modifying the server configuration member for DRDA"
Topic: "Configuring access to distributed databases"
Topic: "Modifying the server configuration member"
Topic: “Configuring Server Event Facility rules for Big SQL"
Topic: “Configuring Server Event Facility rules for dashDB"
Topic: "Configuring access to CA IDMS data"
Topic: "Configuring the server started task JCL"
Topic: "Modifying the server configuration member for CA IDMS"
Topic: "Verifying access to CA IDMS data"

In chapter “Loading data from non-DB2, remote DB2, and remote system sources”:
Topic: "Accelerator Loader server restrictions and considerations"
Topic: "Creating virtual tables for Adabas data"
Topic: "Creating virtual tables for DBMS (DB2) data"
Topic: "Creating virtual tables for RDBMS data"
Topic: "Creating virtual tables for IMS data"
Topic: "Creating virtual tables for VSAM data"
Topic: "Creating virtual tables for sequential data"
Topic: "Creating virtual tables for zFS and HFS file system data"
Topic: "Creating virtual tables for CA IDMS data"
Topic: "Accelerator Loader preferences"
Topic: "SQL preferences"

In chapter “Administering the Accelerator Loader server”:
Topic: "Modifying the data and index buffers for VSAM files"
Topic: "Virtual Parallel Data"
Topic: "Configuring Virtual Parallel Data"
Topic: "Innovation Access Method (IAM)"
Topic: "Configuring MapReduce for IAM"
Topic: "Metadata repository"
Topic: "Populating the metadata repository"

In chapter "Troubleshooting":
Topic: "Messages and codes"

Chapter "Overview"

Topic: “What’s new”
Add the following descriptions:

- The following changes have been made to Tools Customizer:
  - The default size of the global variable file that is used by the Accelerator Loader server has been increased.
  - The following server parameters have been added to the Product Parameters panel:
    - **Enable support for SMF log streams and in-memory resources**
    - **Call the interface module for IAM**
  - The following changes have been made in the HLOIN00 template:
    - Parameters have been added for Virtual Parallel Data support.
    - Parameters have been added for enhanced MapReduce support.
- The following list highlights the enhancements to the Accelerator Loader server:
  - AES 256-bit can now be used to encrypt the password when the driver is establishing a connection with the server.
  - A new composite SMF virtual table rule replaces all existing SMF virtual table rules. The updated member *hlq*.SHLVXVTB(HLVSMFT1) contains all the functionality in the previous SMF virtual table rules HLVSMFT1, HLVSMFT2, HLVSMFT3, HLVSMFT4, and HLVSMFT5.
  - MapReduce and parallelism support is now available for accessing native IMS OSAM files. IMS compression exit support is also included for all supported IMS Direct database types.
  - SQL access to CA IDMS record and set information managed by CA IDMS central versions running on z/OS is now provided. Virtual tables are mapped to IDMS records and sets that can be joined using standard SQL to navigate IDMS information. MapReduce processing is supported to improve performance of large data extracts of IDMS information.
  - The ability to access IBM dashDB data sources via DRDA Virtual Request Facility (VRF) using standard SQL processing is provided. DRDA VRF is a feature that is designed to access data sources which provide the DRDA protocol.
  - IBM Big SQL data sources are now uniquely identified by TYPE(BIGSQL) in the DEFINE DATABASE statement.
  - In the Accelerator Loader studio, when virtualizing DB2/DRDA data sources, the user can select multiple DB2/DRDA tables and run the new wizard once to generate the required virtual tables. Previously, the user was required to run the Virtual Table creation wizard multiple times for each DB2/DRDA source table.
  - The user can now set the data and index buffers for VSAM files. Two new Accelerator Loader server parameters, SQLENGVSAMDATABUFF and SQLENGVSAMINDEXBUFF, have been introduced to control these settings. Previously, the values were hardcoded.
  - Virtual Parallel Data (VPD) now supports Adabas files; VSAM ESDS, KSDS, and RRDS files; and IAM files.
  - IMS Direct now supports Fast Path data entry databases (DEDBs).
  - A batch job with sample DRDARange and IMSRange commands, which are used to populate the Accelerator Loader server metadata repository, is now provided. The job is located in *hlq*.SHLVCNTL(HLVRANGE). Instructions for required edits to the job are provided in the member.
  - MapReduce now supports Innovation Access Method (IAM) files. IAM is a VSAM optimization product distributed by Innovation Data Processing.
  - A new set of SQL preferences has been added to the Accelerator Loader studio. These settings are related to SQL query generation, the SQL Results view, and SQL metadata retrieval. These new settings can improve the performance of metadata retrieval for DB2 and DRDA tables.
  - A new Accelerator Loader server parameter has been introduced which specifies to automatically map all DECFLOAT columns defined in Accelerator Loader server virtual tables to DOUBLE at runtime. DECFLOAT is not a supported data type in the accelerator.

Chapter "Preparing to customize"

Topic: "Worksheets: Gathering parameter values for Tools Customizer"

In subtopic “Task: Create Accelerator Loader files”, the following rows have been updated in the table:

Step or parameter	Required?	Discovered?	Default value	Your value
Global variable file primary allocation Specifies the primary allocation, in cylinders, for the global variable checkpoint data set. Approximately 1180 variables can fit in one cylinder.	Yes	No	250	
Global variable file secondary allocation Specifies the secondary allocation, in cylinders, for the global variable checkpoint data set. Approximately 1180 variables can fit in one cylinder.	Yes	No	50	

In subtopic “Task: Create the server and the server components (required)”, the following rows have been added or updated in the table:

Step or parameter	Required?	Discovered?	Default value	Your value
Enable support for SMF log streams and in-memory resources Specifies whether support for SMF log streams and in-memory resources is enabled. Valid values are YES and NO.	No	No	No	
Call the interface module for IAM Specifies whether to call the interface module for IAM to analyze keys and set ranges for MapReduce. Valid values are YES and NO.	No	No	No	

Chapter “Customizing DB2 Analytics Accelerator Loader”

Update the topics as described below. Topics are located at the end of this document.

Replace the following topics with updated content:
Topic: "IMS data access methods"
Topic: "Configuring access to System Management Facility (SMF) files"
Topic: "Modifying the server configuration member for DRDA"
Topic: "Configuring access to distributed databases"
Topic: "Modifying the server configuration member"

Add the following topics:
Topic: "Modifying the server configuration member for IMS Direct"
Topic: "Configuring access to CA IDMS data"
Topic: "Configuring the server started task JCL"
Topic: "Modifying the server configuration member for CA IDMS"
Topic: "Verifying access to CA IDMS data"
Topic: “Configuring Server Event Facility rules for Big SQL"
Topic: “Configuring Server Event Facility rules for dashDB"

Chapter “Loading data from non-DB2, remote DB2, and remote system sources”

Topic: "Accelerator Loader server restrictions and considerations"
Add the following point under the "Data source and target restrictions, considerations, and limitations" bullet:

- DECFLOAT is not a supported data type in the accelerator. Using Accelerator Loader server parameters, you can specify to automatically map all DECFLOAT columns defined in Accelerator Loader server virtual tables to DOUBLE at runtime. You can also edit the virtual tables in the Accelerator Loader studio, changing the DECFLOAT columns to another data type. For example, data could be converted to DECIMAL(x,x) or CHAR/VARCHAR. The server will then do the necessary conversions and the studio will generate the appropriate load jobs using supported accelerator data types. Therefore, if DECFLOAT, the studio will always generate DDL using a datatype of DOUBLE in the accelerator; otherwise, it will use the specified data type. The server parameters that control this feature are SQLENGDECFLTTODBL and SQLENGDRDATYPECONV. For information about setting these parameters, see "Modifying the server configuration member for DRDA."

Topic: "Creating virtual tables for Adabas data"
Topic: "Creating virtual tables for IMS data"
Topic: "Creating virtual tables for sequential data"
Topic: "Creating virtual tables for VSAM data"
Topic: "Creating virtual tables for zFS and HFS file system data"
Step 2 in each of the following topics will be changed to read:
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.

Remove the following topic:
Topic: "Creating virtual tables for DBMS (DB2) data "

Replace the following topic with updated content:
Topic: "Accelerator Loader preferences"

Add the following new topics:
Topic: "Creating virtual tables for RDBMS data "
Topic: "Creating virtual tables for CA IDMS data"
Topic: "SQL preferences"

Chapter “Administering the Accelerator Loader server”

Update the topics as described below. Topics are located at the end of this document.

Replace the following topics with updated content:
Topic: "Virtual Parallel Data"
Topic: "Configuring Virtual Parallel Data"

Add the following new topics:
Topic: "Modifying the data and index buffers for VSAM files"
Topic: "Innovation Access Method (IAM)"
Topic: "Configuring MapReduce for IAM"
Topic: "Metadata repository"
Topic: "Populating the metadata repository"

Chapter “Troubleshooting”

Topic: “Messages and codes”
Add the following messages.

HLV0265E IDMS support cannot be enabled - module *IDMS-module-name* not found
Explanation: The CA IDMS load module was not found in the server started task JCL.
User response: Add the CA IDMS load libraries into the STEPLIB of the server started task JCL. See “Configuring the server started task JCL” for more information.

IMS data access methods

IMS data can be accessed by the Accelerator Loader server using different data access methods.

By default Accelerator Loader server will access IMS data directly using the underlying VSAM data sets. This access method, called "IMS Direct", provides both map reduce and parallelism support for accessing native IMS files. This support bypasses the requirement of having to use native IMS API calls by reading the IMS database files directly, similar to how an unload utility may work. This method provides a significant increase in performance and reduced elapsed time in processing analytical type queries.

When an IMS SQL query is run, the SQL engine for the server will determine if the request is best executed using IMS Direct (native file support) or if IMS APIs are required. The determination is based on database and file types supported as well as the size of the database. Virtual tables of the IMS segments are required.

The following types of IMS databases are currently supported by IMS Direct:

- Hierarchical direct access method (HDAM) - VSAM and OSAM
- Hierarchical indexed direct access method (HIDAM) - VSAM and OSAM
- Partitioned HDAM (PHDAM) - VSAM and OSAM
- Partitioned HIDAM (PHIDAM) - VSAM and OSAM
- Fast Path data entry database (DEDB)

When using IMS Direct, there is no locking involved when accessing the data, so updates may not be captured and deleted records may have been captured. Security is managed on the IMS native data set itself when IMS Direct is used. The user ID of the client connection must have the necessary security permissions for reading the IMS database data set(s).

When IMS Direct access is not available, the Accelerator Loader server will use DBCTL access using map reduce and parallelism support. Map reduce is an algorithm that enables the Accelerator Loader server to streamline how it accesses IMS data, thereby reducing the processing time required to virtualize IMS data. Statistics about the IMS database are collected and stored within a metadata repository from which the SQL engine optimizes the map reduce process.

In order to exploit the map reduce architecture for IMS using DBCTL as the access method, the Accelerator Loader server must collect information about the IMS database so that it can be used by the SQL engine optimizer. This information is stored within the Accelerator Loader server metadata repository for optimization and can be refreshed at regular intervals.

Metadata repository

The metadata repository for MapReduce stores statistics about virtual tables defined on IMS data sources that are used to enhance performance in conjunction with MapReduce. This support applies to IMS and all DRDA backend data sources, including those accessed via the IBM Federated Server (such as Terradata and Sybase), as well as data sources accessed via the Accelerator Loader server's direct DRDA support (DB2 LUW and Oracle).

This information can be collected by the following command query:

```
SELECT IMSRange('IMS DBD name')
```

The following sample batch job can be executed at regular intervals to populate the IMS metadata repository with fresh statistics. This sample job is provided in *hlq*.SHLVCNTL(HLVRANGE). Instructions for required edits to the job are provided in the member.

```
//RANGE EXEC PGM=HLVXMAPD,PARM='SSID=hlvid',MXR=30000000'
//STEPLIB  DD DISP=SHR,DSN=hlq.SHLVLOAD
//RPT      DD SYSOUT=*
//FMT      DD SYSOUT=*,DCB=LRECL=4096
//IN       DD *
 SELECT IMSRANGE('<IMS DBD NAME>');
/*
```

where:

- *hlvid* is the name of the Accelerator Loader server started task that was customized using Tools Customizer
- *hlq*.SHLVLOAD is the Accelerator Loader server load library
- *IMS DBD Name* is the four-character IMS subsystem name.

No additional configuration or customization is required to take advantage of either of these access methods.

Modifying the server configuration member for IMS Direct

To optionally configure IMS Direct, configure IMS Direct parameters in your Accelerator Loader server configuration file.

Procedure

1. In the *hlvid*IN00 member, locate the comment "Enable IMS Direct Map Reduce."
2. Add the following statements:

   ```
   "MODIFY PARM NAME(IMSDIRECTCYLBUF) VALUE(3)"
   "MODIFY PARM NAME(IMSDIRECTOSAMRECSRD) VALUE(2)"
   ```

   The following table lists the parameters for configuring support for IMS Direct:

Parameter	Description	Valid values
IMSDIRECTCYLBUF	Specifies the number of cylinders of data to buffer for each file processed in an IMS Direct task.	1-50. Default value is 3.
IMSDIRECTOSAMRECSRD	Specifies the number of records to read in each OSAM I/O operation. Note that for random reads, a large number may lead to unnecessary blocks read. For sequential reads, small numbers may give decreased performance.	1-50. Default value is 2.

Configuring access to System Management Facility (SMF) files

By default, access to System Management Facility (SMF) files is enabled in the Accelerator Loader server started task JCL and the server configuration member. To enable reading SMF data real-time using log streams, you must have the **SMFPRMxx** member in the system PARMLIB data set configured to use both log streams and in-memory resources. Follow the steps in this section to use SMF GDG data set names, or to use dynamic data set names.

About this task

SMF data set names are dynamic in local environments and require SEF rules enablement and optionally Global Variables set to specific values to provide data set names to the virtual tables and views when using SMF data set or log stream configurations.

You can choose either GDG data set name support or dynamic data set name support, or both, to quickly access your SMF data. These two options are provided for your convenience to help you start accessing your SMF data. Custom rules may need to be developed to use your local naming convention to access your SMF files.

Procedure

1. To enable real-time access to SMF data, add the following statements to the *hlvid*IN00 member after the `GLOBAL PRODUCT OPTIONS` statement.

```
IF DoThis
  THEN DO
    "DEFINE SMF NAME(IFASMF.INMEM)",
    "BUFSIZE(500)",
    "TIME(0)"
END
```

   Note: You must have the **SMFPRMxx** member in the system PARMLIB data set configured to use log streams and in-memory resources.

Parameter	Description	Valid values
BUFSIZE	Indicates how much SMF data (megabytes) will be retained in memory for queries. If the buffer fills up, the oldest data will be discarded. In parallel, SMF is recording these records to a logstream.	1-10,000
TIME	Indicates how long (in minutes) to keep SMF data in memory. Older data will be discarded. Specifying 0 indicates no time limit and data will be retained until the buffer fills up.	0-999

2. Enable reading SMF data from GDG data sets and access to SMF data using dynamic data set names by enabling Server Event Facility rule HLVSMFT1 in the VTB ruleset. You can select from a GDG data set, any SMF dump data set, a log stream data set, or the in-memory stream. You can choose to activate all three options by customizing the rule.
   a. Use the following steps to enable rule HLVSMFT1 in the VTB ruleset:
      1) On the main menu, select **Server administration**.
      2) In the Administer Accelerator Loader Server menu, specify option `3`, **Manage Rules**.
      3) Specify option `2`, **SEF Rule Management**.
      4) Enter `VTB` for **Display Only the Ruleset Named**.
      5) Enable the rule by specifying `E` and pressing Enter.
      6) Set the rule to Auto-enable by specifying `A` and pressing Enter.

         Setting the rule to Auto-enable activates the rule automatically when the server is re-started.
   b. Configure the access method using one or more of the following methods:
      - Review the information in the rule for the instructions on setting Global Variables that will be used by the rule. Navigate one screen back on the ISPF panel, or start over by going to option `3`, **Manage Rules**, and then option `1`, **Global Variables**. In the Global Variables display, perform the following steps:
        1) Change Global Prefix to `GLOBAL2`.
        2) Select SMFTBL2 by entering `S` next to the SMFTBL2 data set.
        3) Configure the SMF data access option. DEFAULT should have corresponding SMF dump data set names if used. This option can be used to specify the source SMF, such as GDGBASE, INMEM, and LOGSTREAM.

        Note:

        VTB rules and global variables may be used to reference a GDG data set, any SMF dump data set, a log stream data set, or the in-memory stream. For example:

        ```
        GLOBAL2.SMFGBL2.YESTERDAY = "YOUR.DATASET.SMFDUMP(-1)"
        GLOBAL2.SMFGBL2.M2 = "YOUR.DATASET.SMFDUMP(-2)"
        GLOBAL2.SMFGBL2.M3 = "YOUR.DATASET.SMFDUMP(-3)"
        GLOBAL2.SMFGBL2.M4 = "YOUR.DATASET.SMFDUMP(-4)"
        GLOBAL2.SMFGBL2.M5 = "YOUR.DATASET.SMFDUMP(-5)"
        GLOBAL2.SMFGBL2.IM = "IFASMF.INMEM"
        GLOBAL2.SMFGBL2.IM2 = "IFASMF.INMEM2"
        GLOBAL2.SMFGBL2.LOG = "LOGSTREAM.dataset.name"
        ```
      - Pass a dynamic data set name for SMF tables using the following format for the table name in the SQL statement:

        `TableMapName__DataSetName`

        Where `DataSetName` is prefixed by two underscores (__) and the periods in the data set name are replaced with single underscores (_).

        For example, `SELECT * FROM SMF_01400__DATA_SET_NAME` would translate into an SQL query of `SELECT * FROM SMF_14000` and access the data set `DATA.SET.NAME`.
      - Pass a dynamic data set name for SMF virtual views using the following format for the virtual view name in the SQL statement:

        `ViewMapName__DataSetName`

Where `DataSetName` is prefixed by two underscores (__) and the periods in the data set name are replaced with single underscores (_).

For example, `SELECT * FROM SMFV_01400__DATA_SET_NAME` would translate into an SQL query of `SELECT * FROM SMFV_01400` and access the data set `DATA.SET.NAME`.

Modifying the server configuration member for DRDA

If you are using a zIIP specialty engine, enable the RDBMS access method for Distributed Relational Database Architecture (DRDA) in the server configuration member.

About this task

Configure the server to use Distributed Relational Database Architecture (DRDA) when accessing a RDBMS.

The server configuration member *hlvid*IN00 is in data set *hlq*.SHLVEXEC, where *hlvid* represents the name of the Accelerator Loader server started task that was customized using Tools Customizer.

Procedure

1. Verify that the Unicode translation of the Coded Character Set Identifier (CCSID) used in the `DEFINE DATABASE` statement and the CCSID used by the target RDBMS are defined for your z/OS environment.
   a. You should identify the CCSID of the RDBMS.

      For example, Oracle may use *ccsid1*. In your `DEFINE DATABASE` statement in the configuration member for the RDBMS you have *ccsid2*. For this example, where Oracle is using *ccsid1*, you need to verify that you have *ccsid1-ccsid2* and *ccsid2-ccsid1* defined in your Unicode translation table on z/OS using the command **D UNI,ALL**.
   b. If the entry is not present, you need to add the entry to your Unicode translation table and refresh.

      Please refer to the IBM z/OS documentation on how to add the entry.

      Note: As an alternative, the Unicode table can be appended within the server by using the following statement examples in the server configuration member:

      ```
      "DEFINE CONV SOURCE(ccsid1) TARGET(ccsid2) TECH(RE)"
      "DEFINE CONV SOURCE(ccsid2) TARGET(ccsid1) TECH(RE)"
      ```
2. In the *hlvid*IN00 member, locate the section that contains the comment `Enable DRDA access to DB2 database subsystems`.
3. Enable the DRDA parameters by changing the syntax `if DontDoThis` to `if DoThis`, and then set the `DRDASKIPZSERVICES` parameter to `YES`. The following example shows the section in the configuration member to enable:

   ```
   /*-------------------------------------------------------------------*/
   /* Enable DRDA access to DB2 database subsystems                     */
   /*-------------------------------------------------------------------*/
   if DoThis then
     do
     "MODIFY PARM NAME(TRACEOEDRDARW)        VALUE(YES)"
     "MODIFY PARM NAME(CLIENTMUSTELECTDRDA)  VALUE(NO)"
     "MODIFY PARM NAME(DRDASKIPWLMSETUP)     VALUE(NO)"
     "MODIFY PARM NAME(DRDAFORLOGGINGTASK)   VALUE(NO)"
     "MODIFY PARM NAME(DRDASKIPZSERVICES)    VALUE(YES)"
   ```

   The following table describes these parameters:

Parameter	Description	Valid values
TRACEOEDRDARW	If set to YES (recommended), TCP/IP communications via DRDA are traced. If set to NO, DRDA receive and send operations are not traced.	**YES** **NO** Default value.
CLIENTMUSTELECTDRDA	If set to YES, JDBC clients must explicitly opt in for DRDA to be used by setting the user parameter connection variable to 'DRDA'. **Note:** JDBC clients can always opt out of DRDA processing by setting the user parameter to 'NODRDA'. If set to NO, DRDA processing is used for access all configured RDBMSs.	**YES** **NO** Default value.
DRDASKIPWLMSETUP	If set to YES, WLM information is not collected and sent to DRDA during JDBC logon processing. If captured, the DRDA equivalent to SET_CLIENT_ID calls is issued after logon to establish these values on the DRDA connection. If not captured, the transmission that is used to set these WLM-related values is bypassed. If set to NO, the client user ID, application name, workstation name, and accounting token that were sent in the initial client buffer are collected and sent separately after logon processing to DRDA.	**YES** **NO** Default value.
DRDAFORLOGGINGTASK	If set to YES, DRDA processing is used for the DB2 on z/OS logging subtask. If set to NO, SAF or RRSAF connections are used. **Note:** Passticket support must be enabled for the target DDF server. If passticket support is not configured, set the parameter to NO.	**YES** **NO** Default value.

Parameter	Description	Valid values
DRDASKIPZSERVICES	Prevents DRDA from being used for z/Service DB2 processing. If set to YES, z/Services client tasks do not use DRDA processing for DB2 requests. If set to NO, DRDA will be used when configured for a particular DB2 connection. **Note:** Passticket support must be enabled for all target DDF servers.	**YES** **NO** Default value.

4. If you will need to map DECFLOAT columns defined in Accelerator Loader server virtual tables to DOUBLE, add the following statements:

```
"MODIFY PARM NAME(SQLENGDECFLTTODBL)    VALUE(YES)"
"MODIFY PARM NAME(SQLENGDRDATYPECONV)   VALUE(YES)"
```

   The following table describes these parameters.

   Note: For more information about this feature, see Accelerator Loader server restrictions and considerations.

Parameter	Description	Valid values
SQLENGDECFLTTODBL	Forces translation of DECFLOAT fields to DOUBLE (long hex float). You can override this option using a virtual table rule. This option will convert inbound DECFLOAT columns to DOUBLE (hex float long). The data will still be presented as DECFLOAT in the metadata. In a virtual table rule, set OPTBDRDF to `Y` to enable the conversion, or `N` to disable it. Any other value in OPTBDRDF will be ignored, and the global setting will be used.	**YES** **NO** Default value.
SQLENGDRDATYPECONV	Allow data type conversions for DRDA columns. This option allows the data type in the map to be different from the actual data type. When this occurs, the SQL engine will convert the data, and the metadata will reflect the data type in the map. You can override this option using a virtual table rule.	**YES** **NO** Default value.

5. Define DRDA RDBMSs by entering a definition statement. Provide your local environment values for all the parameters. The following example shows the section in the configuration member to enable:

```
"DEFINE DATABASE TYPE(type_selection)"        ,
                 "NAME(name)"                 ,
                 "LOCATION(location)"         ,
                 "DDFSTATUS(ENABLE)"          ,
                 "DOMAIN(your.domain.name)"   ,
                 "PORT(port)"                 ,
                 "IPADDR(1.1.1.1)"            ,
                 "CCSID(37)"                  ,
                 "APPLNAME(DSN1LU)"           ,
                 "IDLETIME(110)"              ,
```

Where *type_selection* is either `GROUP`, `MEMBER`, or `ZOSDRDA`.

The following table lists the parameters for defining DDF endpoints:

Parameter	Description	Valid values
APPLNAME	Application name. The APPLNAME used by the target endpoint for passticket generations. (*Optional*)	A valid value is 1 - 8 characters. If APPLNAME is not specified in the definition statement, no default value is provided and passticket access is disabled. **Note:** APPLNAME is not required when connecting from the JDBC driver.
AUTHTYPE	Authentication type. This can be either DES for Diffie Hellman Encryption Standard or AES for Advanced Encryption Standard. When AUTHTYPE is not supplied the default is DES. To force AES the option must be added to the DEFINE DATABASE, each server can be different in what is supported as to AES/DES.	**DES** Diffie Hellman Encryption Standard (default value) **AES** Advanced Encryption Standard.
CCSID	Specify the EBCDIC single-byte application CCSID (Coded Character Set Identifier) configured for this RDBMS subsystem on the RDBMS installation panel DSNTIPF, option 7. (*Optional*)	Refer to the RDBMS vendor documentation for a list of valid CCSID.
DDFSTATUS	The DDF activation status can be altered online by using the ISPF 4-DB2 dialog panels. (*Required*)	**ENABLE** To make this DDF definition active within Accelerator Loader server. **DISABLE** DDF endpoint is not used.

Parameter	Description	Valid values
DOMAIN	The part of a network address that identifies it as belonging to a particular domain.	No default value.
IPADDR	Specify the dot-notation IPV4 address of the DDF endpoint. (*Optional*)	If this parameter is not specified, the value 127.0.0.1 (local host) is the default. For group director definitions, use the DVIPA IP address of the group director.
LOCATION	For DB2: The DB2 location name. For LUW: The LUW database. For Oracle: The Oracle SSID as defined to the Oracle Database Provider (Gateway) (*Required*)	A valid value is a string 1 - 16 characters.
NAME	The database name as known to the server. (*Required*)	A valid value consists of 1 - 4 characters. Clients use this ID when they request access to a specific DB2 subsystem.
PORT	The TCP/IP port at which the server is listening. (*Required*)	If this keyword is not entered, the default DRDA port number 443 is used.
SECMEC	The DRDA security mechanism in force for standard dashDB services requires an authentication method setting. Define as either USRENCPWD, which informs the server to encrypt the PASSWORD or EUSRIDPWD, which informs the server to encrypt the USERID and PASSWORD during the initial connection to dashDB. (*For GROUP and MEMBER types.*)	**USRENCPWD** Encrypt password only. **EUSRIDPWD** Encrypt userid and password.

Parameter	Description	Valid values
TYPE	For DB2 for z/OS: **GROUP** If this DDF endpoint is a DB2 group director, specify GROUP. **MEMBER** If this DDF endpoint is a DB2 instance or group member for z/OS, specify MEMBER. **ZOSDRDA** If this DDF endpoint is a remote LPAR access with SEF ATH rule processing, typically used for connecting to a DB2 on another LPAR that uses a different passwords as the local LPAR specify ZOSDRDA. An ATH rule may be necessary to manage the credentials. ZOSDRDA allows for ATH rule management of credentials.	For DB2 for z/OS: GROUP MEMBER ZOSDRDA

Configuring access to distributed databases

You can configure access to data on Big SQL, dashDB, DB2 LUW (Linux, UNIX, and Windows), Microsoft SQL Server, Oracle, and QMF DRDA.

Before you begin

If you are connecting to a Big SQL or DB2 LUW database, then you must install and configure the IBM DB2 Federated Server. For additional information, refer to the documentation on the IBM website.

If you are connecting to an Oracle database, then you must install and configure the Oracle Database Provider for DRDA. For additional information, refer to the documentation on the Oracle website.

If you are connecting to a 2016 Microsoft SQL Server database, then you must install and configure the Host Integration Server for HIS DRDA Service. For additional information, refer to the documentation on the Microsoft website. The SYSIBM Views from Microsoft must be installed.

About this task

Configure access to distributed databases by modifying the configuration member, configuring Server Event Facility (SEF) rules, and optionally setting up alternate authentication information.

Procedure

1. "Modifying the server configuration member."
2. Configure the Server Event Facility rules and set up authentication for the appropriate database.
   - "Configuring Server Event Facility rules for Big SQL" on page 19.
   - "Configuring Server Event Facility rules for dashDB" on page 21.
   - Configuring Server Event Facility rules for Linux, UNIX, and Windows.
   - Configuring Server Event Facility rules for Microsoft SQL Server.
   - Configuring Server Event Facility rules for Oracle DRDA.
   - Configuring Server Event Facility rules for QMF DRDA Server.

Modifying the server configuration member

Configure the Accelerator Loader server to access data sources using Distributed Relational Database Architecture (DRDA).

About this task

The Accelerator Loader server is enabled for DRDA access. To access data sources using DRDA, modify the Accelerator Loader server parameter member *hlvid*IN00 that was configured using Tools Customizer, and define those data sources to the configuration member.

The server configuration member *hlvid*IN00 is in data set *hlq*.SHLVEXEC, where *hlvid* represents the name of the Accelerator Loader server started task that was customized using Tools Customizer.

Procedure

1. Verify that the Unicode translation of the Coded Character Set Identifier (CCSID) used in the `DEFINE DATABASE` statement and the CCSID used by the target RDBMS are defined for your z/OS environment.
   a. Identify the CCSID of the RDBMS.

      For example, Oracle may use *ccsid1*. In your `DEFINE DATABASE` statement in the configuration member for the RDBMS you have *ccsid2*. For this example, where Oracle is using *ccsid1*, you need to verify that you have *ccsid1-ccsid2* and *ccsid2-ccsid1* defined in your Unicode translation table on z/OS using the command **D UNI,ALL**.
   b. If the entry is not present, add the entry to your Unicode translation table and refresh.

      Refer to the IBM z/OS documentation on how to add the entry.

      Note: As an alternative, the Unicode table can be appended within the Accelerator Loader server by using the following statement examples in the server configuration member:

      ```
      "DEFINE CONV SOURCE(ccsid1) TARGET(ccsid2) TECH(RE)"
      "DEFINE CONV SOURCE(ccsid2) TARGET(ccsid1) TECH(RE)"
      ```
2. In the *hlvid*IN00 member, locate the section that contains the comment "Enable DRDA access to DB2 database subsystems."
3. Define DRDA RDBMSs by entering a definition statement. Provide your local environment values for all the parameters.

   ```
   "DEFINE DATABASE TYPE(type_selection)"         ,
                    "NAME(name)"                  ,
                    "LOCATION(location)"          ,
                    "DDFSTATUS(ENABLE)"           ,
                    "DOMAIN(your.domain.name)"    ,
                    "PORT(port)"                  ,
                    "IPADDR(1.1.1.1)"             ,
                    "CCSID(37)"                   ,
                    "APPLNAME(DSN1LU)"            ,
                    "IDLETIME(110)"               ,
   ```

   This is an example for dashDB:

   ```
   "DEFINE DATABASE TYPE(DASHDB)"                ,
                    "NAME(name)"                 ,
                    "LOCATION(location)"         ,
                    "AUTHTYPE(AES)"              ,
                    "SECMEC(EUSRIDPWD)"          ,
                    "DDFSTATUS(ENABLE)"          ,
                    "DOMAIN(your.domain.name)"   ,
                    "PORT(port)"                 ,
                    "CCSID(37)"                  ,
   ```

   The following table lists the parameters for defining DDF endpoints:

Parameter	Description	Valid values
TYPE	For distributed databases: **BIGSQL** DDF endpoint is a Big SQL engine. **DASHDB** DDF endpoint is a dashDB database. **LUW** DDF endpoint is a DB2 instance or group member for Linux, UNIX, or Windows. **MSSQL** DDF endpoint is a DB2 instance or group member for Microsoft SQL Server. **ORACLE** DDF endpoint is an Oracle instance. The parameter informs DRDA AR and supportive tooling that the remote server is an Oracle Database Provider which supports DRDA AS. The Oracle DRDA AS must be in z/OS simulation mode. **QMFDRDA** DDF endpoint is a QMF DRDA AS Object Server instance.	For distributed databases: BIGSQL DASHDB LUW MSSQL ORACLE QMFDRDA
NAME	The database name as known to the server. (*Required*)	A valid value consists of 1 - 4 characters. Clients use this ID when they request access to a specific DB2 subsystem.

Parameter	Description	Valid values
LOCATION	For DB2: The DB2 location name. For dashDB: This is the database name of the dashDB database or alias name for the database. For LUW: The LUW database. For Oracle: The Oracle SSID as defined to the Oracle Database Provider (Gateway). (*Required*)	A valid value is a string 1 - 16 characters.
DDFSTATUS	The DDF activation status can be altered online by using the ISPF 4-DB2 dialog panels. (*Required*)	**ENABLE** Make this DDF definition active. **DISABLE** DDF endpoint is not used.
DOMAIN	The part of a network address that identifies it as belonging to a particular domain. Either DOMAIN or IPADDR is required, but not both.	No default value.
PORT	The TCP/IP port at which the server is listening. (*Required*)	A valid 1-5 numeric string.
IPADDR	Specify the dot-notation IPV4 address of the DDF endpoint. Either DOMAIN or IPADDR is required, but not both.	If this parameter is not specified, the value 127.0.0.1 (local host) is the default. For group director definitions, use the DVIPA IP address of the group director.
CCSID	Specify the EBCDIC single-byte application CCSID (Coded Character Set Identifier) configured for this RDBMS subsystem on the RDBMS installation panel DSNTIPF, option 7. (*Optional*)	Refer to the RDBMS vendor documentation for a list of valid CCSIDs.
APPLNAME	Application name. The APPLNAME used by the target endpoint for passticket generations. (*Optional*)	A valid value is 1 - 8 characters. If APPLNAME is not specified in the definition statement, no default value is provided and passticket access is disabled. **Note:** APPLNAME is not required when connecting from the JDBC driver.

Parameter	Description	Valid values
IDLETIME	If DB2 ZPARM parameter IDTHTOIN is set to a non-zero value set IDLETIME to a value slightly lower (10 secs.) than IDTHTOIN. This will also allow product DRDA threads to become inactive. (*DB2 for z/OS only*)	0-9999 seconds.
AUTHTYPE	Authentication type. This can be either DES for Diffie Hellman Encryption Standard or AES for Advanced Encryption Standard. When AUTHTYPE is not supplied, the default is DES. To force AES, the option must be added to the DEFINE DATABASE statement. Each server can be different in what is supported as to AES/DES.	**DES** Diffie Hellman Encryption Standard (default value) **AES** Advanced Encryption Standard.

Configuring Server Event Facility rules for Big SQL

Configure Server Event Facility (SEF) rules to provide access to Big SQL databases.

Procedure

1. Auto-enable the SQL rule SHLVXSQL(HLVSBIGC) to allow Accelerator Loader studio Meta discovery on Big SQL databases.
   a. On the Administer Accelerator Loader Server menu, select option **3** for Manage Rules.
   b. Select option **2** for SEF Rule Management.
   c. Enter `*` to display all rules, or `SQL` to display only SQL rules.
   d. Set Auto-Enable for the HLVSBIGC rule member by entering `A` and pressing Enter.
2. Auto-enable the SEF ATH rule SHLVXATH(HLVABIGG) to provide the logon credentials to each Big SQL instance. Global variables are used to define alternate authentication credential mapping for the SEF ATH rule.
   a. On the Administer Accelerator Loader Server menu, select option **3** for Manage Rules.
   b. Select option **2** for SEF Rule Management.
   c. Enter `*` to display all rules, or `ATH` to display only authentication rules.
   d. Set Auto-Enable for the HLVABIGG rule member by entering `A` and pressing Enter.
3. Optional: Verify the HLV global variable setup for authentication rules:
   a. On the Administer Accelerator Loader Server menu, select option **3** for Manage Rules.
   b. Select option **1** for Global Variables.
   c. Enter `GLOBAL2` and press enter to display all GLOBAL2 Variables.

d. If subnode DRDA does not exist, enter `S DRDA` in the command line and press Enter.
e. If subnode ATH does not exist, enter `S ATH` in the command line and press Enter.
f. If subnode HLVABIGG does not exist, enter `S HLVABIGG` in the command line and press Enter.

Example

It is common for data centers to assign different user IDs for access to z/OS and for access to Big SQL. By default, the server will attempt to log on to Big SQL with the same user ID that was presented for logon to z/OS. A facility is needed in the server to optionally change a user's logon credentials when accessing DB2 on Big SQL.

A Server Event Facility (SEF) ATH rule can be used to set these two parameters for logon to Big SQL:

```
ATH.AUDROTUS     Big SQL User ID
ATH.AUDROTPW     Big SQL Password
```

A SEF rule could be coded to insert hard coded user IDs and passwords, even testing for different incoming IDs and specific Big SQL databases. However, as an alternative to hard coding IDs and passwords in the SEF rule, *hlq*.SHLVXATH(HLVABIGG) is provided to resolve Big SQL credentials from the Server Global Variables. Rule HLVABIGG should not be modified, except to toggle tracing of its execution. Server Global Variables for Big SQL are mapped as follows:

```
GLOBAL2.DRDA.ATH.REXX.GLOBAL.DEFAULT
GLOBAL2.DRDA.ATH.REXX.GLOBAL.userid
GLOBAL2.DRDA.ATH.REXX.ssid.GLOBAL.DEFAULT
GLOBAL2.DRDA.ATH.REXX.ssid.userid
```

Where:

- *REXX* is the name of the active SEF ATH rule, in this case HLVABIGG.
- *ssid* is the target Big SQL subsystem name in the `xLVyIN00 DEFINE DATABASE NAME(ssid)` statement.
- *userid* is the incoming z/OS user ID.

The last node of each Global Variable sets the Big SQL user ID and password in the format:

```
userid:password;comment after the semicolon
```

For instance, using active SEF rule HLVABIGG, a default Big SQL user ID and password could be set to `biguser/bigpswd` by setting the subnode value of `GLOBAL2.DRDA.ATH.HLVABIGG.GLOBAL.DEFAULT` to:

```
biguser:bigpswd;Global Default for any user
```

Similarly, a default Big SQL user ID and password for subsystem BIG1 could be set to `big1user/big1pswd` by setting the subnode value of `GLOBAL2.DRDA.ATH.HLVABIGG.BIG1.GLOBAL.DEFAULT` to:

```
big1user:big1pswd;Default for Big SQL subsystem named BIG1
```

Each user can be assigned unique credentials for `BIG1`. Assuming *ZOSUSER1* and *ZOSUSER2* are the z/OS user IDs:

```
GLOBAL2.DRDA.ATH.HLVABIGG.BIG1.ZOSUSER1
subnode value
big1usera:pswda;BIG1 credentials for ZOSUSER1
GLOBAL2.DRDA.ATH.HLVABIGG.BIG1.ZOSUSER2
subnode value
big1userb:pswdb;BIG1 credentials for ZOSUSER2
etc.
```

Each user can be assigned unique default credentials for all Big SQL databases. Assuming *ZOSUSER1* and *ZOSUSER2* are the z/OS user IDs:

```
GLOBAL2.DRDA.ATH.HLVABIGG.GLOBAL.ZOSUSER1
subnode value
big1usera:pswda;BIG credentials for ZOSUSER1
GLOBAL2.DRDA.ATH.HLVABIGG.GLOBAL.ZOSUSER2
subnode value
big1userb:pswdb;BIG credentials for ZOSUSER2
etc.
```

Searches to resolve the BIG credentials follow the order:

```
GLOBAL2.DRDA.ATH.HLVABIGG.ssid.userid
GLOBAL2.DRDA.ATH.HLVABIGG.GLOBAL.userid
GLOBAL2.DRDA.ATH.HLVABIGG.ssid.GLOBAL.DEFAULT
GLOBAL2.DRDA.ATH.HLVABIGG.GLOBAL.DEFAULT
```

Global Variables are maintained between recycles of the Accelerator Loader server and between IPLs.

Configuring Server Event Facility rules for dashDB

Configure Server Event Facility (SEF) rules to provide access to IBM dashDB databases.

Procedure

1. Auto-enable the SQL rule SHLVXSQL(HLVSDDBC) to allow Accelerator Loader studio Meta discovery on dashDB databases.
   a. On the Administer Accelerator Loader Server menu, select option **3** for Manage Rules.
   b. Select option **2** for SEF Rule Management.
   c. Enter `*` to display all rules, or `SQL` to display only SQL rules.
   d. Set Auto-Enable for the HLVSDDBC rule member by entering `A` and pressing Enter.
2. Auto-enable the SEF ATH rule SHLVXATH(HLVADDBG) to provide the logon credentials to each dashDB instance. Global variables are used to define alternate authentication credential mapping for the SEF ATH rule.
   a. On the Administer Accelerator Loader Server menu, select option **3** for Manage Rules.
   b. Select option **2** for SEF Rule Management.
   c. Enter `*` to display all rules, or `ATH` to display only authentication rules.
   d. Set Auto-Enable for the HLVADDBG rule member by entering `A` and pressing Enter.
3. Optional: Verify the HLV global variable setup for authentication rules:
   a. On the Administer Accelerator Loader Server menu, select option **3** for Manage Rules.
   b. Select option **1** for Global Variables.
   c. Enter `GLOBAL2` and press enter to display all GLOBAL2 Variables.

d. If subnode DRDA does not exist, enter `S DRDA` in the command line and press Enter.
e. If subnode ATH does not exist, enter `S ATH` in the command line and press Enter.
f. If subnode HLVADDBG does not exist, enter `S HLVADDBG` in the command line and press Enter.

Example

It is common for data centers to assign different user IDs for access to z/OS and for access to dashDB. By default, the server will attempt to log on to dashDB with the same user ID that was presented for logon to z/OS. A facility is needed in the server to optionally change a user's logon credentials when accessing dashDB.

A Server Event Facility (SEF) ATH rule can be used to set these two parameters for logon to dashDB:

```
ATH.AUDROTUS     DashDB ID
ATH.AUDROTPW     DashDB Password
```

A SEF rule could be coded to insert hard coded user IDs and passwords, even testing for different incoming IDs and specific dashDB databases. However, as an alternative to hard coding IDs and passwords in the SEF Rule, *hlq*.SHLVXATH(HLVADDBG) is provided to resolve dashDB credentials from the Server Global Variables. Rule HLVADDBG should not be modified, except to toggle tracing of its execution. Server Global Variables for dashDB are mapped as follows:

```
GLOBAL2.DRDA.ATH.REXX.GLOBAL.DEFAULT
GLOBAL2.DRDA.ATH.REXX.GLOBAL.userid
GLOBAL2.DRDA.ATH.REXX.ssid.GLOBAL.DEFAULT
GLOBAL2.DRDA.ATH.REXX.ssid.userid
```

Where:

- *REXX* is the name of the active SEF ATH rule, in this case HLVADDBG.
- *ssid* is the target dashDB subsystem name in the `xLVyIN00 DEFINE DATABASE NAME(ssid)` statement.
- *userid* is the incoming z/OS user ID.

The last node of each Global Variable sets the dashDB user ID and password in the format:

```
userid:password;comment after the semicolon
```

For instance, using active SEF Rule HLVADDBG, a default dashDB user ID and password could be set to `ddbuser/ddbpswd` by setting the subnode value of `GLOBAL2.DRDA.ATH.HLVADDBG.GLOBAL.DEFAULT` to:

```
ddbuser:ddbpswd;Global Default for any user
```

Similarly, a default dashDB user ID and password for subsystem DDB1 could be set to `ddb1user/ddb1pswd` by setting the subnode value of `GLOBAL2.DRDA.ATH.HLVADDBG.DDB1.GLOBAL.DEFAULT` to:

```
ddb1user:ddb1pswd;Default for DDB subsystem named DDB1
```

Each user can be assigned unique credentials for `DDB1`. Assuming *ZOSUSER1* and *ZOSUSER2* are the z/OS user IDs:

```
GLOBAL2.DRDA.ATH.HLVADDBG.DDB1.ZOSUSER1
subnode value
DDB1usera:pswda;DDB1 credentials for ZOSUSER1
```

```
GLOBAL2.DRDA.ATH.HLVADDBG.DDB1.ZOSUSER2
subnode value
DDB1userb:pswdb;DDB1 credentials for ZOSUSER2
etc.
```

Each user can be assigned unique default credentials for all dashDBs. Assuming *ZOSUSER1* and *ZOSUSER2* are the z/OS user IDs:

```
GLOBAL2.DRDA.ATH.HLVADDBG.GLOBAL.ZOSUSER1
subnode value
DDB1usera:pswda;DDB credentials for ZOSUSER1
GLOBAL2.DRDA.ATH.HLVADDBG.GLOBAL.ZOSUSER2
subnode value
DDB1userb:pswdb;DDB credentials for ZOSUSER2
etc.
```

Searches to resolve the dashDB credentials follow the order:

```
GLOBAL2.DRDA.ATH.HLVADDBG.ssid.userid
GLOBAL2.DRDA.ATH.HLVADDBG.GLOBAL.userid
GLOBAL2.DRDA.ATH.HLVADDBG.ssid.GLOBAL.DEFAULT
GLOBAL2.DRDA.ATH.HLVADDBG.GLOBAL.DEFAULT
```

The following panel shows an example of dashDB Global Variables for the ATH rule HLVADDBG:

```
-------------------------- Display Global Variables ------ Row 1 to 10 of 10

 Command ===> Scroll ===> CSR
 LCs: S Show Subnodes M Modify Value X Hex Browse
 D Remove Node and Subnodes P Drop Node B Browse

Global Prefix: GLOBAL2.DRDA.ATH.HLVADDBG

S Subnode Name Nodes Subnode Value
-------------- ----- -------------------------------------------------------
 EXMP 0 TYPE(dashDB) dashDB DRDA Provider Example NAME(EXMP)
 GLOBAL 2 TYPE(dashDB) GLOBAL DEFAULT ATH USERID/PASSWORD
```

The following panel shows an example of dashDB Global Variables for SUBSYS NAME(EXMP) to swap USERID to ALTUSER:

```
-------------------------- Display Global Variables -------- Row 1 to 1 of 1
 Command ===> Scroll ===> CSR
 LCs: S Show Subnodes M Modify Value X Hex Browse
 D Remove Node and Subnodes P Drop Node B Browse

Global Prefix: GLOBAL2.DRDA.ATH.HLVADDBG.EXMP

S Subnode Name Nodes Subnode Value
-------------- ----- -------------------------------------------------------
 USERID 0 ALTUSER:PASSWORD;USERID SWAP TO ALTUSER
```

Configuring access to CA IDMS data

To access CA IDMS data, you must configure the Accelerator Loader server started task JCL. You can then optionally verify access to the data.

Accelerator Loader server started task JCL changes are required to access CA IDMS software and define default CA IDMS settings.

Restrictions

The following restrictions and considerations apply when accessing CA IDMS data:

- CA IDMS releases v18.0 and newer are supported.
- Virtual table mapping is only supported through the Accelerator Loader studio. No batch utilities or ISPF interfaces are provided to map tables.
- Access is limited to a single CA IDMS central version from each Accelerator Loader server.
- All database access is in central version mode. Single user (local mode) is not supported.
- All access is read only. SQL update is not supported.
- CA IDMS Logical Record Facility (LRF) is not supported.
- Data access uses CA IDMS network DML only. Access to databases defined using the CA IDMS SQL product is not supported.
- IDMS VSAM Transparency is not supported.
- Map reduce client support is limited to database areas using 24-bit page IDs.
- SQL queries run under the z/OS user credentials associated with the client connection. Users must have authority to bind CA IDMS run units and access subschema information per site security standards.

Note:

Server configuration parameters control the following behaviors and can be modified if necessary:

- CA IDMS run-unit management, specifically maximum run-units and a timeout value for inactive run-units
- CA IDMS access tracing

Configuring the server started task JCL

Modify the server started task JCL to access CA IDMS and define default CA IDMS settings.

Before you begin

All data sets that you add to the server started task JCL STEPLIB must be APF-authorized.

About this task

Modify the server started task JCL to access CA IDMS and define default IDMS settings.

Procedure

1. Add the CA IDMS load libraries to the STEPLIB, which are required for CA IDMS central version access.
2. Add the SYSCTL DD statement identifying the CA IDMS central version to access.
3. Add the SYSIDMS statement with additional environment parameters. Minimally, this data set should include a CVRETRY=OFF statement to prevent an WTOR message when the CA IDMS central version is not active.
4. Add the CA IDMS system message data set to DCMSG.

Modifying the server configuration member for CA IDMS

To optionally configure server parameters for CA IDMS, you can update your Accelerator Loader server configuration file.

About this task

The CA IDMS server parameters can assist you in configuring CA IDMS data access. In most typical environments, the default settings for these parameters will not need modification.

Procedure

1. In data set *hlq*.SHLVEXEC, locate member *hlvid*IN00, where *hlvid* represents the name of the Accelerator Loader server started task that was customized by using Tools Customizer.
2. Add the following statements to your *hlvid*IN00 member:

   ```
   "MODIFY PARM NAME(MAXIDMSRUNUNITS) VALUE(4)"
   "MODIFY PARM NAME(SQLENGIDMSRUTIMOUT) VALUE(60)"
   ```

   The following table lists the parameters for configuring CA IDMS data access:

Parameter	Description	Valid values
MAXIDMSRUNUNITS	Maximum IDMS max units. This parameter limits the number of concurrent IDMS run units that a server will start to access a CA IDMS central version. Limiting concurrent IDMS run units will prevent storage related user 3134 abends when creating run units with CA IDMS.	Positive numeric value. Default value is 4.
SQLENGIDMSRUTIMOUT	CA IDMS run unit inactivity timeout. Specifies the length of time in seconds to keep a run unit active for reuse by subsequent SQL queries in a client connection.	Positive numeric value. Default value is 60 seconds.

Verifying access to CA IDMS data

To verify access to CA IDMS data, you can optionally install a set of maps to the sample database EMPDEMO and run queries using the installed maps.

Before you begin

The CA IDMS sample database EMPDEMO must be installed in the central version you plan to access.

About this task

You can customize and run the provided IVP job HLVISIV1 to install maps to the EMPDEMO database and network schema maps to the SYSTEM database.

The following maps are installed for verification testing using the sample EMPDEMO database:

Table 1. CA IDMS EMPDEMO database maps

Map	Description
EMPSS01_EMPLOYEE	Enables SQL access to EMPLOYEE record.
EMPSS01_OFFICE	Enables SQL access to the OFFICE record.
EMPSS01_DEPARTMENT	Enables SQL access to the DEPARTMENT record.
EMPSS01_OFFICE_EMPLOYEE	Enables SQL access to the OFFICE-EMPLOYEE set for joining the EMPSS01_OFFICE and EMPSS01_EMPLOYEE tables.
EMPSS01_DEPT_EMPLOYEE	Enables SQL access to the DEPT-EMPLOYEE set for joining the EMPSS01_DEPARTMENT and EMPSS01_EMPLOYEE tables.

The network schema maps can be used for verification purposes if the EMPDEMO database is not installed in your central version. These maps access records and sets in the CA IDMS network schema IDMSNTWK, providing SQL access to application metadata. The following table provides a subset of the installed network schema maps that can be used for verification purposes:

Table 2. CA IDMS network schema IDMSNTWK maps

Map	Description
IDMSNWKA_S_010	Enables SQL access to the S-010 network schema record. S-010 records describe application schemas defined to your IDMS central version.
IDMSNWKA_SS_026	Enables SQL access to the SS-026 network schema record. SS-026 records describe application subschemas defined to your IDMS central version.
IDMSNWKA_SSR_032	Enables SQL access to the SSR-032 network schema record. SSR-32 records describe application subschema records defined to your IDMS central version.
IDMSNWKA_S_SS	Enables SQL access to the S-SS set for joining the IDMSNWKA_S_010 and IDMSNWKA_SS_026 tables.
IDMSNWKA_SS_SSR	Enables SQL access to the SS-SSR set for joining the IDMSNWKA_SS_026 and IDMSNWKA_SSR_032 tables.

Procedure

1. Locate the HLVISIV1 member in the *hlq*.SHLVCNTL data set.
2. Modify the JCL according to the instructions provided in the HLVISIV1 member.
3. Submit the job.
4. If the server is active, use the following instructions to refresh maps and make the maps available for use:
   a. From the Primary Option Menu, specify option `D`, **Data Mapping**, and press Enter.
   b. From the Data Mapping Facility menu, specify option `3`, **Map Refresh**, and press Enter.

Results

HLVISIV1 installs CA IDMS EMPDEMO and network schema maps into the server map data set.

Creating virtual tables for RDBMS data sources

Create virtual tables that map to RDBMS data sources, such as DB2 on z/OS, DB2 LUW (Linux, UNIX, and Windows), Oracle, and Microsoft SQL Server.

Before you begin

It is recommended that you create a virtual table for each RDBMS table from which you want to access data. Creating a virtual table for each RDBMS table allows you to perform joins across data that may originate from different DRDA accessible RDBMS subsystems or to perform joins between your RDBMS data and other types of virtualized data, such as IMS or VSAM data.

This wizard allows you to create multiple virtual tables at a time, if the selected source tables belong to the same RDBMS subsystem. In this wizard, a view is treated the same as a table; each table or view is mapped to a virtual table.

Procedure

1. On the **Server** tab, explore the RDBMS metadata information by expanding the **SQL** > **Data** > **Other Subsystems** node, and then navigating down the appropriate subtree. The hierarchy begins with the subsystem, followed by the schema, and then the tables and views.
2. Select a single table or view from the tree, or use the following techniques to select multiple tables or views:
   - To select more than one individual node, hold down the Ctrl key and click each node to be included.
   - To select a range of tables (or views), click the first table in the range, and then hold the Shift key and select the last table in the range. All tables within the range will be included.
   - To select a group of nodes, click the parent node. All of the children under the parent node will be included. For example, select the **Tables** node to include all tables belonging to that schema. Or, select the schema node to include all tables and views under that schema.

   You can use a combination of these techniques. For example, you can select two schema nodes to create virtual tables for all tables and views belonging to those two schemas.
3. Right-click the selected items and select **Create Virtual Table(s)**. The **New Virtual Tables Wizard** launches.
4. On the **New Virtual Tables for DBMS access** page, complete the following fields:

Field	Action
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.

Field	Action
Naming Patterns	Specify the format to use for the generated virtual table names. Use the following variables to create naming patterns that are derived from the RDBMS metadata: • {Subsystem}: Subsystem name • {Schema}: Source schema 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 RDBMS 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 RDBMS subsystem that is referenced in the virtual tables. For example, you create a virtual target system called TSDSN1, and specify that it will access the RDBMS subsystem DSN1. Then, you create 50 virtual tables that access data in the RDBMS source TSDSN1 (that is, pointing to DSN1). If it becomes necessary to change the name of the RDBMS source DSN1, 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.
Advanced	When reading large volumes of data from tables, click **Advanced** to display and configure the **MapReduce** feature. The **MapReduce** feature enables you to divide the data into logical partitions and process those partitions in parallel using the **Thread Count** value. At runtime, the number of zIIP processors is verified and one thread is used for each zIIP processor; resulting in improved performance. The **Thread Count** value you specify overrides the default value (2) and the discovered value. To disable **MapReduce**, select the **Disable MapReduce** check box.

5. 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.
6. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Generating and executing SQL queries.

Creating virtual tables for CA IDMS data

Create virtual tables that map to the CA IDMS data that you want to access and from which the SQL used to access the data is generated and executed.

Before you begin

The Accelerator Loader server must be configured for CA IDMS access, and the CA IDMS central version referenced by the data server SYSCTL DD statement must be active.

About this task

CA IDMS schema records are mapped using the CA IDMS data dictionary. Each record is mapped as a separate virtual table using the COBOL names to derive the SQL column names. In addition to records, schema sets can be mapped as well. CA IDMS set virtual tables serve as correlation tables between CA IDMS records so SQL joins can navigate the CA IDMS schema.

Procedure

1. On the **Server** tab, explore the CA IDMS metadata information by expanding the **Discovery** > **IDMS** node, and then navigating down the appropriate subtree. The hierarchy begins with the data dictionary, followed by the CA IDMS schema, the CA IDMS subschema, and then the associated records and sets.
2. Select one or more records, as follows:
   - To select individual records, hold down the Ctrl key and click each record to include.
   - To select a range of records, click the first record in the range, and then hold the Shift key and select the last record in the range. All records within the range will be included.
3. Right-click the selected records and select **Create Virtual Table(s)**. The **New Virtual Tables Wizard** launches.

   Note: You can map the relevant CA IDMS sets in the wizard.
4. On the **Create IDMS virtual tables** page, complete the following **Common Virtual Table Settings**:

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.

Field	Description
Arrays Handling	Select one of the following options: • **Flatten arrays into a single fixed table at runtime (Y)**: This option supports both OCCURS and OCCURS DEPENDING ON statements. • **Return arrays into separate tables at runtime (N)**: This option supports both OCCURS and OCCURS DEPENDING ON statements. A subtable is generated for each array. Subtables support SQL read access only.
Virtual Table Naming Patterns	Specify the format to use for the generated virtual table names. You can specify different patterns for records and sets. Use the following variables to create naming patterns that are derived from the IDMS metadata: • {SubSchema}: Subschema name • {Record}: Record name • {Set}: Set name
Prune IDMS record field suffix from column names	Select this option to remove the IDMS record field suffix from the column names.

5. In the table that lists the IDMS records, 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.
6. To map the sets, click **Fetch Related IDMS Sets**. The Accelerator Loader studio collects additional metadata from the server and displays the relevant items in the table that lists the IDMS sets.
7. In the table that lists the IDMS sets, review the list of selected entries. Modify the selections as needed.
8. To disable MapReduce, click **Advanced** and select **Disable MapReduce**.
9. Click **Finish**.

Results

The Accelerator Loader studio creates the virtual tables (the metadata maps) on the server.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Generating and executing SQL queries.

Accelerator Loader preferences

Use **Accelerator Loader** preferences to set preferences such as general session and SQL results settings.

General **Accelerator Loader** preferences are identified and described in the table that follows.

Field	Description
Enable Server Tracing of Studio Calls	Includes the Accelerator Loader studio trace calls in your server trace results. This setting is disabled by default.
Studio HTTP Debug Option	The Accelerator Loader studio type of debug option to be used. The default setting is **Normal**.
Studio Fixed Width Font	Determines the font, font style, and font size that displays in Accelerator Loader studio. The default setting is **Courier New-regular-9**.
Hex Encoding	Sets the Hex encoding to use. The default setting is **UTF-8**.
File Encoding	Determines the file encoding setting to use. The default setting is **windows-1252**.
CSV File Delimiter	Determines the type of file delimiter to use for CSV files. The default setting is a comma (,).
New Connection (DSN) Naming Pattern	Determines the naming pattern to use when new connections are made. The default setting is **{SubSystem}**.
Studio Connection Timeout (secs)	The number of seconds to wait before a server connection is determined to be unsuccessful. The default setting is **10**.
Studio Operation Timeout (secs)	The number of seconds to wait before determining that the Accelerator Loader studio operation is unsuccessful. The default setting is **30**.
Studio Remote Control Port	The port number that the Accelerator Loader studio uses for remote connections. The default setting is **31416**.
Use UPPER case logon credentials for both JDBC and HTTP connections	Select this check box to require that logon credentials use uppercase characters for JDBC driver and HTTP connections. This setting is enabled by default. For systems that have mixed-case password support, you must clear this check box and add the following statement to your *hlq*`.SHLVEXEC(hlvidIN00)` file: `"MODIFY PARM NAME(PASSWORDCASE) VALUE(ASIS)"`

SQL preferences

Use **SQL** preferences to specify settings related to SQL query generation, the SQL Results view, and SQL metadata retrieval.

SQL preferences are identified and described in the table that follows.

Field	Description
SQL Generate Query Behavior	Determines whether you are prompted to execute SQL or if SQL executes automatically. Options include: • **Generate query and issue user prompt**. This is the default setting. • **Generate and execute query (no prompt)** • **Generate query but do not execute query (no prompt)**
SQL Results Max Rows	Specifies an upper limit on the number of rows that will be displayed per query in the **SQL Results** view. The default value is **1000**.
SQL Results Max Bytes	Specifies an upper limit on the number of data bytes that will be displayed per query in the **SQL Results** view. The default value is **1000000**.
SQL Results values accessed as	Specifies how data values are returned. Options include **String** or **Object**. The default setting is **String**.
Use prepared statement to retrieve SQL column info for DB2 or DRDA tables	The Accelerator Loader studio obtains column metadata information from the server for DB2 and DRDA tables and views when you expand a table or view node under the **Other Subsystems** tree in the **Server** view, or in other situations where column information needs to be retrieved. The Accelerator Loader studio supports two different ways of retrieving this column metadata information: • Using a prepared statement. Typically, this server call will be faster; however, this option requires that the user have SELECT privileges to the table in the remote database. This method is the default and will be used when this preference is selected. • Using the JDBC getColumns() API. This method is the more conventional approach; however, in some cases (for example, Oracle), the remote DRDA subsystem may take a long time to process the metadata query. This method will be used when this preference is cleared.

Field	Description
Fetch primary key and index information for virtual tables	If this preference is selected, then when you expand a virtual table or view in the **Server** view, any primary key or indexed column nodes will be identified. This identification process requires the Accelerator Loader studio to make additional metadata calls to the server. To disable these calls and the associated identifications, you can clear this preference and thus speed up the time taken to populate the column nodes. This preference is selected by default.
Fetch primary key and index information for DB2 or DRDA tables	If this preference is selected, then when you expand a table or view node under the **Other Subsystems** tree in the **Server** view, any primary key or indexed column nodes will be identified. This identification process requires the Accelerator Loader studio to make additional metadata calls to the server (and subsequently to the remote database). In some cases, these additional calls may be rather expensive (for example, Oracle). To disable these calls and the associated identifications, you can clear this preference to speed up the time taken to populate the column nodes. This preference is cleared by default.

Modifying the data and index buffers for VSAM files

You can change the data and index buffers for VSAM files.

About this task

To control the settings of the data and index buffers for VSAM files, you must add the required parameters to you Accelerator Loader server configuration file.

Procedure

1. In data set *hlq*.SHLVEXEC, locate member *hlvid*IN00, where *hlvid* represents the name of the Accelerator Loader server started task that was customized by using Tools Customizer.
2. Add the following statements to your *hlvid*IN00 member:

   ```
   "MODIFY PARM NAME(SQLENGVSAMDATABUFF) VALUE(20)"
   "MODIFY PARM NAME(SQLENGVSAMINDEXBUFF) VALUE(30)"
   ```

   The following table lists these parameters:

Parameter	Description	Valid values
SQLENGVSAMDATABUFF	Specifies the number of data buffers for VSAM files. Default: 20	Numeric value.
SQLENGVSAMINDEXBUFF	Specifies the number of index buffer for VSAM files. Default: 30	Numeric value.

MapReduce

This section provides information on MapReduce features for performance enhancement.

You should also refer to the *IBM® DB2® Analytics Accelerator Loader 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 Accelerator Loader 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:

- Physical sequential data sets on disk, tape, or virtual tape
- Log streams
- Adabas files
- VSAM KSDS, RRDS, and ESDS files
- IAM files
- zFS/HFS files

Configuring Virtual Parallel Data

To configure Virtual Parallel Data, optionally configure VPD parameters in your Accelerator Loader server configuration file. To use VPD when loading data, specify a group name and appropriate parameters when generating your load JCL.

Procedure

1. Configure the following parameters in the *hlvid*IN00 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.
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 in the **Generate JCL to Load Accelerator** wizard in the Accelerator Loader studio.
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 Accelerator Loader 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 hlvidIN00 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 hlvidIN00 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 Accelerator Loader server.

Before you begin

The Accelerator Loader server must already be installed. Use these instructions to configure the Accelerator Loader server.

About this task

To enable MapReduce for IAM, the Accelerator Loader server configuration file must be configured. Customizing this member is done using Tools Customizer.

Procedure

1. Invoke Tools Customizer for z/OS.
2. Access the Product Parameters panel.
3. Under the task **Create the server and the server components**, select the steps **Create the server** and **Create the server parameters**, and provide a value for the following field:

Step or parameter	Required?	Discovered?	Default	Your value
Call the interface module for IAM Specifies whether to call the interface module for IAM to analyze keys and set ranges for MapReduce. Valid values are YES and NO.	No	No	No	

4. Generate the customization jobs. The jobs are based on the HLOHLVS and HLOIN00 templates. For more information, see Generating customization jobs.
5. Submit the customization jobs. For more information, see Submitting customization jobs.

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 Accelerator Loader server. The gathered metadata persists across server restarts.

Populating the metadata repository

You can periodically run the **DRDARange** or **IMSRange** command to gather metadata repository information about the backend virtual tables.

About this task

You can run the metadata repository command for DRDA or IMS 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.
  - For the IMS data source, use the SELECT statement at the database level.

    ```
    SELECT IMSRANGE('IMS database name')
    ```
- 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*.SHLVCNTL(HLVRANGE). Instructions for required edits to the job are provided in the member.

  ```
  //RANGE  EXEC  PGM=HLVXMAPD,PARM='SSID=hlvid,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',...);
|                          SELECT IMSRANGE('<IMS DBD Name>');

|
```