Open Data Analytics for z/OS
Version 1 Release 1

User's Guide

SC27-9034-00

Note

Before using this information and the product it supports, read the information in "Notices" on page 65.

This edition applies to Version 1 Release 1 of IBM® Open Data Analytics for z/OS® (5655-OD1) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2019-02-18

Contents

About this information

This information supports IBM Open Data Analytics for z/OS (5655-OD1) and contains information about the Data Service Studio, which is a component that is provided with IBM Open Data Analytics for z/OS.

Purpose of this information

This document presents the information you need to get SQL access to your mainframe data using the Data Service Studio.

Who should read this information

This information is intended for system and database administrators.

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

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

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
: If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM Knowledge Center function
: If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
: If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

  To help us better process your submission, include the following information:

  - Your name, company/university/institution name, and email address
  - The following deliverable title and order number: Open Data Analytics for z/OS User's Guide, SC27-9034-00
  - The section title 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 provided for sending documentation comments. Instead, take one or more of the following actions:

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

Summary of changes for IBM Open Data Analytics for z/OS User's Guide

The following changes are made to Version 1 Release 1.

March 2019

- *Db2 Virtualization* is a new feature that provides single-point access to various data source types. See "Creating Db2 user-defined table functions" on page 33 and "SQL preferences" on page 59.
- When using SMF log streams, you can use the LS_TIMESTAMP and LS_TIMESTAMP_LOCAL virtual columns to retrieve timestamp values. When used in a WHERE predicate, the timestamp is searched using the respective time zone. See "System Management File sample code" on page 45.

June 2018

- Db2 Direct is a new Data Service server access method used to access Db2 data by reading the data in the underlying Db2 VSAM linear data sets directly. The Db2 data access method is specified when creating virtual tables for access to Db2 data. See "Creating virtual tables for RDBMS data sources" on page 13.
- The process of creating maps to access VSAM and sequential data has been simplified by support of the following methods:
  - Querying information in the IBM Application Discovery and Delivery Intelligence (ADDI) dictionary. See "Creating virtual tables for VSAM and sequential access using ADDI" on page 29.
  - Querying information in the IBM Rational Asset Analyzer (RAA) dictionary. See "Creating virtual tables for VSAM and sequential access using RAA" on page 31 and "Metadata Discovery preferences" on page 59.

April 2018

- When connecting from the Data Service Studio to the Data Service server, password phrase authentication is supported. See "Connecting to the Data Service server" on page 5.
- IMS Direct now supports access to multiple IMS subsystems. See "Creating virtual tables for IMS data" on page 16, which includes updated procedures.
- SQL access to IBM MQ is now provided. See "Creating virtual tables for IBM MQ" on page 21.
- You can now specify a generation data group base name when defining a virtual table. See "Creating virtual tables for sequential data" on page 22.
- SQL query access to DB2 unload data sets is now provided. See Chapter 8, "Accessing DB2 unload data," on page 47.

Chapter 1. Data Service Studio overview

This topic introduces the Data Service Studio and the function it provides in Mainframe Data Service.

Data Service enables users to have seamless access to mainframe data without needing to know technical details, such as how the data is formatted or where it is located. Data Service provides data integration without the complexity and cost associated with extracting, transforming, and loading the data.

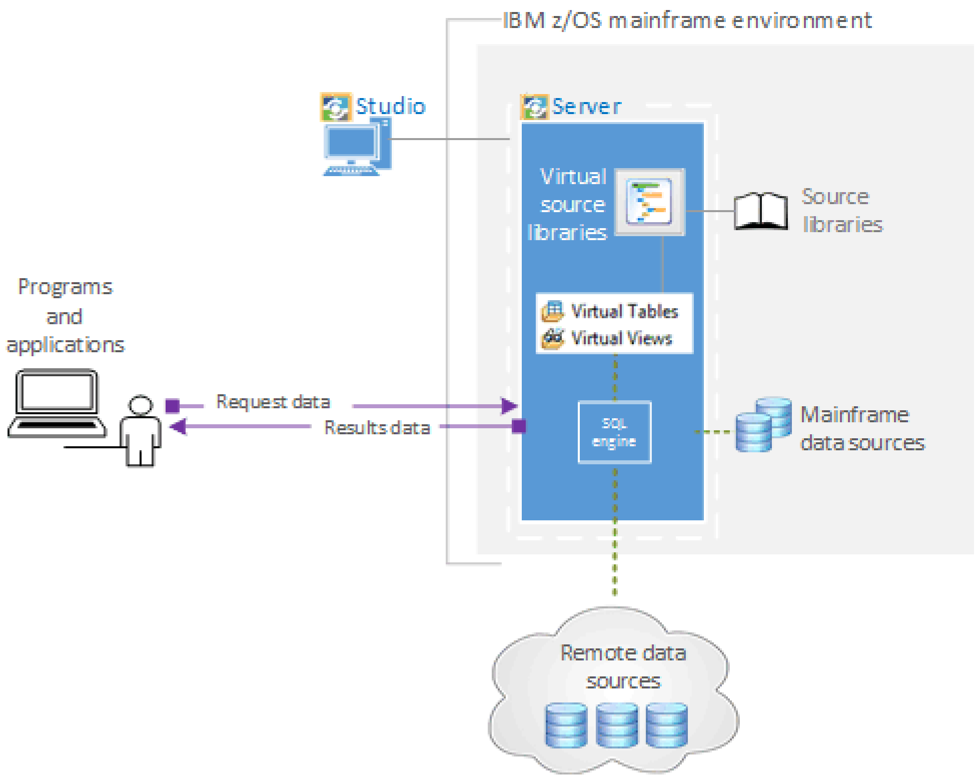
The following mainframe data sources are supported by Data Service:

- Adabas
- Db2 for z/OS
- CA IDMS
- IMS
- IBM MQ
- Sequential
- VSAM, VSAM CICS, and IAM
- zFS and HFS

Support is also provided for Oracle, Microsoft SQL Server, and Db2 for LUW data sources. In addition, you can access the systems data in System Management Files (SMF), System Log (SysLog) files, and Operational Log (OperLog) files, from which you can perform IT Operational Analytics (ITOA).

Data Service Studio is an Eclipse-based user interface that communicates with the Data Service server. You use the studio to create metadata objects, such as virtual source libraries, virtual tables and virtual views on the host Data Service server. These server metadata objects provide the information that is necessary to access your mainframe data and off-host RDBMS data sources by virtualizing your data.

The illustration that follows shows the main components within the Mainframe Data Service architecture.



Getting started with the studio

The following list highlights the steps to start using DS Studio:

- Before you can begin using the DS Studio, you must first open the Data Service perspective from the **Window** menu, and then connect to the Data Service server on the mainframe that has the data that you want to access.
- To get access to the mainframe data, use the DS Studio to create the following components on the Data Service server:
  - *Virtual source libraries*: A virtual source library is a reference to a library that already exists on the mainframe. Virtual source libraries point to the information (metadata) that is required to virtualize the source data. You create virtual source libraries using the **Virtual Source Library Wizard**.
  - *Virtual tables*: A *virtual table* is a map to the data that you want to access from the data source. After the virtual table is created, use it to generate and execute the SQL. The resulting SQL is used to read and extract the mapped data from the mainframe. You create virtual tables using the **New Virtual Table Wizard**. To get access to your data from programs or applications, you generate the code from the SQL using the **SQL Code Wizard**.
  - *Virtual views*: Optionally, you can use virtual tables to create virtual views from which you can generate SQL queries. A virtual view is the SQL SELECT statement that contains the columns from the source data that are used to read data directly from the data source. In some cases, creating a virtual view is more convenient than regenerating and editing the SQL. Virtual views are also used if your virtual table is missing columns or if you want to join different types of data. You create virtual views using the **New Virtual View Wizard**.

Chapter 2. Perspectives

The perspective that you choose determines the views and editors that become available in the workbench.

A perspective is an arrangement of views and editors in the workbench. You use perspectives to accomplish a specific task or set of tasks. When you open a perspective, the menu items, tool bars, views, editors, and wizards that are associated with that perspective become available in the workbench.

Opening perspectives

From the **Window** menu, you can open a perspective by selecting **Open Perspective** and selecting the perspective that you want to use from the drop-down menu.

Data Service perspective

The **Data Service** perspective provides the default views, editors, and wizards that you use to perform tasks that are associated with accessing your mainframe data.

Use this perspective to perform the following tasks:

- Explore mainframe resources and view metadata.
- Create and manage data sources.
- Generate and modify SQL queries.
- Create virtual tables from SQL.
- Create virtual views for use with complex SQL queries.
- Generate the code to use in your applications from SQL.

Views

The following views are available with this perspective:

Views	Description
Active Connections	Lists the open JDBC connections between the Data Service Studio and one or more servers. The current active connection is used by the SQL Editor to issue SQL queries over that JDBC connection. You can create new or delete existing server connections.
Server view	Lists data resources, stored procedures, and metadata. You can perform tasks on selected objects in the tree. Explorer views include the following tabs: • **Client**: Lists information that is related to data sources and application development on your local machine. • **Server**: Lists the server to which you want to connect, view resources, or perform tasks. • **Network**: Lists the host and server connections within your network. You can choose to view or modify existing host and server connection settings.

Views	Description
	• **Favorites**: Lists shortcuts to the mainframe resources that you frequently access.
Server Trace	Applies labels to Server Trace messages for use when searching within the **Server Trace** view.
Lists	Use to display details for each tree node or object that is selected in an **Explorer** view.
Search	Use to search for a text string within the Server Trace results.
Server Trace	Use to set and gather server diagnostic information for support purposes.
Server Trace Import	Use to import Server Trace (`.isx`) files.
SQL Results	Use to display the result set returned from a SQL query in the **SQL Results** tab, and the resulting trace information in the **SQL Messages** tab.
Studio Navigator	Use to list shortcuts to key task views, wizards, and editors.
Virtualization Facility	Use to display virtual table mapping details.

Editors

The following text editors are available with this perspective:

Editors	Description
Data Source	Use to edit existing connection definitions.
SQL	Use to compose SQL statements and to invoke queries against the server.
Virtualization Facility	Use to edit metadata settings related to virtual tables and virtual views.

Wizards

This perspective includes wizards that guide you through tasks, such as:

- Setting the server connection.
- Creating virtual source libraries.
- Creating virtual tables for SQL access to data.
- Generating application code from SQL.

Chapter 3. Connecting to the Data Service server

To access data on the mainframe, connect Data Service Studio to the Data Service server that is running on an z/OS mainframe instance.

Connecting to the Data Service server

Use the Data Service Studio to connect to the Data Service server that is running on an instance of z/OS.

Before you begin

Before you can connect to the Data Service server, the server must be configured and started.

Procedure

1. From the Data Service Studio menu, click **Window** > **Open Perspective** > **Data Service**.
2. On the **Server** tab, click **Set Server**.
3. In the **Set Server** dialog box, complete the following:
   - **Host**: Select or enter the TCP/IP host name or IP address of the mainframe system on which the server is deployed.
   - **Port**: Enter the port number that the server uses. The default is 1200.
   - **Userid**: Enter your mainframe user ID.
   - **User Password**: Enter your password or password phrase for the mainframe user ID.
4. Click **OK**.

Completing the configuration of DRDA access to RDBMS data sources

To complete the configuration of DRDA access to RDBMS data sources, you must bind packages on the server, and grant users the authority to use those packages.

Before you begin

You must know the host name and the port number of the server and your log on credentials. Your log on credentials must have the authority to bind packages and grant privileges.

About this task

Perform the following task for each RDBMS data source that you want to access.

Procedure

1. From the Data Service Studio, click **Window** > **Open Perspective** > **Data Service**.
2. On the **Server** tab, click **Set Server**.
3. In the **Set Current Server** dialog box, complete the following fields:

Option	Description
Host	Enter the TCP/IP host name or IP address of the mainframe system.
Port	Enter the port number that is used to communicate with the server. The default is 1200.
Userid	Enter the mainframe user ID.

Option	Description
User Password	Enter the password for the mainframe user ID.

4. Click **OK**.
5. On the **Server** tab, expand **SQL** > **Data** > **Other Subsystems**.
6. Right-click the subsystem and select **BIND/GRANT Packages**.
7. On the **BIND/GRANT Packages** page, complete the following fields:

Field	Action
Package Prefix	Enter the two character prefix to assign to the package. The package prefix must match the prefix that is defined on the mainframe server. If you change the default prefix (DS), you must also change it in the *hlq*`.SAZKEXEC(AZKSIN00)` file.
Number of Cursors	Enter the number of cursors to use to process results. The default is 200.
Collection	Enter the value to use to bind packages. The default is **NULLID**. This value is normally determined by the DB2 Administrator.
Table Qualifier	Enter the value to use to qualify unqualified SQL. This value is normally determined by the DB2 Administrator.
Owner UserId	Enter the user ID of the package owner. This value is normally determined by the DB2 Administrator.
Grant to	Set only when granting authority for the target DB2 server. The default is **PUBLIC**.
Bind Package	Binds the product packages. This is the default setting.
Grant Execute	Grants execute permissions on the package to the user ID that is specified in the **Grant to** field.
Replace Packages	Replaces an existing package for the specified subsystem. Select this option only if the package already exists.

8. Depending on the options that you select, additional dialog boxes and messages might be displayed.
9. Review the results in the **Results** text box and click **BIND/GRANT**.

Locale considerations

You can modify the data source connection definitions to use different local code pages.

Before you begin

You have the option to change the default code page (`US/English IBM 1047`) that the Data Service Studio uses to perform character data translations between the native Java character encoding (UTF-8) and the mainframe.

Procedure

1. To configure the data source connection definition, in the **Active Connections** view, close all open connections.
2. On the **Client** tab, expand **Data Service** > **Data Sources** > **JDBC** > **Default Config File**.
3. Right-click the data source that you want to modify and click **Edit**.
4. In the **Data Source Editor**, click the **Connection String** tab.
5. Add or modify the `Charset` setting to use the appropriate EBCDIC code page. For example, `Charset=IBM037`.
6. If `LGID=ENC` exists in the connection string, delete it to avoid a conflict with the `Charset` setting.

7. Close the **Data Source Editor**.
8. When prompted, click **Yes** to save the data source definition.
9. To change the default `Charset` that the Data Service Studio uses when creating connection definitions, from the **Window** menu select **Preferences**, expand **Data Service** > **Driver**.
10. In **Connection Overrides**, enter the new `Charset` setting and click **OK**.
11. On the **Server** tab, expand **SQL** > **Data**.
12. Right-click the data source to which you want to connect and select **Create Connection Definition (DSN)**.
13. Accept the default name that is displayed or enter a new DSN name and click **OK**.
14. In the **Data Source Editor**, click the **Connection String** tab and confirm that the new `Charset` setting displays in the connection string.

Results

When running queries using the new data source definition, the character data (including language specific glyphs) that you chose is displayed in the **SQL Results** view.

Chapter 4. Creating server metadata

Using the Data Service Studio, you create the server metadata that provides the information necessary to virtualize your data. Server metadata includes virtual source libraries, virtual tables, and virtual views.

Creating virtual source libraries

Virtual source libraries point to the information that Mainframe Data Service needs in order to access some types of mainframe data.

Before you begin

A virtual source library is a server metadata object that references a source library that exists on the Data Service (host) server. The members of the source library contain layout information specific to a type of data, for example a COBOL or PL/I copybook (copybook), Adabas Data Definition Module (DDM) views, IMS Database Definition (DBD) files, or IMS Program Specification Block (PSB) files. Virtual source libraries provide a reusable catalog of the host's data source libraries.

Note: When creating a virtual source library, the current user must have read access to the host data source library.

About this task

Virtual source libraries are a prerequisite to creating virtual tables for the following types of data sources:

- Adabas
- IMS
- IBM MQ
- Sequential
- VSAM, VSAM CICS and IAM
- zFS and HFS

When creating the virtual source libraries you specify the following data set (PDS/PDSE) names based on the type of data that you want to access:

- To access Adabas data, you specify the name of the PDS/PDSE that contains the Data Definition Module (DDM) views that have been set up for the Adabas data in your environment.
- To access IMS data, you may need to create multiple virtual source libraries that reference multiple types of source libraries. You may create a separate virtual source library that references the IMS DBD files, the IMS PSB files, and the copybooks that describe the layout of each IMS segment. In each case, you specify the PDS/PDSE that is specific to the source library.
- To access IBM MQ data, you specify the name of the PDS/PDSE that contains the copybook that describes the data written to the queue.
- To access sequential data, you specify the name of the PDS/PDSE that contains the copybook that describes the structure of the sequential data records.
- To access VSAM, VSAM CICS, and IAM data, you specify the name of the PDS/PDSE that contains the copybook that describes the structure of the VSAM, VSAM CICS, and IAM data records.
- To access z/FS and HFS data, you specify the name of the PDS/PDSE that contains the copybook that describes the structure of the records in the data file.

Procedure

1. On the **Server** tab, expand **Admin** > **Source Libraries**.

2. Right-click **Create Virtual Source Library** and select **Create Virtual Source Library**.
3. On the **Virtual Source Library** page, complete the following fields:

Field	Action
Name	Enter a unique, meaningful name for the virtual source library you are creating.
Description	Enter an optional description for the virtual source library.
Library Name	Enter the name of the PDS/PDSE that contains the layout information for the data you want to access.

4. Click **Finish**.

Results

The new virtual source library is displayed in the **Source Libraries** folder.

Creating virtual tables

To access your data, create a virtual table or virtual view that maps to your source data and that matches the definition of the source data structure on the mainframe.

From the virtual table or virtual view, you generate the SQL that is used to read and access the mapped data from the mainframe. You create virtual tables using the **New Virtual Table Wizard** that is specific to the type of data that you want to access. Some virtual tables, including SMF virtual tables, are made available during the product installation.

The following high-level procedures must be completed prior to creating a virtual table:

- Start the Data Service Studio.
- Open the Data Service perspective.
- Connect to the Data Service server.
- Run the **Create Source Library** wizard to create a virtual source library to map to your mainframe data. This procedure is not required to create virtual tables for RDBMS data.

Virtual table tasks

When a virtual table is selected on the **Server** tab, you can perform the following tasks:

- **Edit**: Edit the virtual table properties in the editor.
- **Copy** and **Paste**: Copy the virtual table and paste the copy under the **Virtual Tables** node.
- **Disable**: Disable the virtual table on this server.
- **Delete**: Delete the virtual table from the server.
- **Create Virtual View**: Create a virtual view from the virtual table.

Key and index information

To view a summary of key and index information for an existing virtual table, select the virtual table on the **Server** tab and from the **Window** menu, select **Show View** > **Properties**. The properties for the selected table are displayed in the **Properties** view.

If a virtual table includes columns that have a primary key or an index, the column is notated using the following symbols:

- Key symbol – This column is associated with a primary key.
- Superscript numeral 1 – This column is associated with a unique index, but does not have an associated primary key.
- Superscript asterisk – This column is associated with a non-unique index.

This primary key and index information is also highlighted when you browse RDBMS tables under the **Other Subsystems** tree.

You can control the identification of primary keys and indexes using settings in SQL preferences.

Creating virtual tables for Adabas data

Create a virtual table that maps to the Adabas data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

Have the Adabas database ID and password, the file number, and the subsystem name available.

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **ADABAS** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
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.
Arrays Handling	Enable one of the following array management options: • **Flatten arrays into a single fixed table at runtime**: This relates to multiple occurring (MU) fields and periodic (PE) groups. • **Return arrays into separate tables at runtime**: This relates to multiple occurring (MU) fields and periodic (PE) groups. A subtable is generated for each array. Subtables only support read access.

5. On the **ADABAS Details** page, complete the following fields and click **Next**:

Field	Action
DB ID	Enter the Adabas database ID.
File Number	Enter the number of the file to use.
Adabas Password	If the file is password-protected, enter the password. This password is stored and encrypted in the virtual table so that future queries use the same password to access the data.
SubSystem	Enter the name of the Adabas subsystem.
Max MU Count	Enter the maximum number of times to repeat the MU field. The default is 10.
Max PE Count	Enter the maximum number of times to repeat the PE field. The default is 10.
Create Count Field	Select this check box to index every MU or PE field so that the index (count) field created precedes the repeating field. This index field tells the caller how many repeating fields are being used.
Secure	Select this check box to choose the Adabas file ID number to be used for file name security.

Field	Action
DE Search only	Select this check box if you want the utility to generate control definitions that allow the client to only use WHERE columns that are Adabas descriptors (such as superde, subde, and hyperde).
Search by PE index	Select this check box to allow the client to target rows that match a particular occurrence of the PE field when searching rows using the WHERE clause. If this parameter is not specified, all rows where any occurrence of that PE field match the value specified will be targeted.
Unpacked to Packed	Select this check box to convert all unpacked format fields to packed format.
Binary to Integer	Select this check box to convert all 2-byte and 4-byte binary fields to short integer and integer formats, respectively.
Binary to Packed	Select this check box to map the binary fields in the Adabas file to SQL decimal columns (numeric packed decimal format) in the generated virtual table. Note the following points: • If the precision of the Adabas binary field allows for the possibility of a numeric value that would cause data overflow when converted to SQL decimal, the column in the virtual table will be mapped to SQL binary instead. This means that Adabas fields with precision greater than 12 will continue to be mapped to SQL binary. • If you select the **Binary to Integer** check box and the **Binary to Packed** check box, the precision of the Adabas binary field will determine if it gets mapped to an SQL integer (that is, 2-byte or 4-byte fields) or a decimal type.
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.

6. Optional: On the **Data Definition Module** page, if you have a Natural Data Definition Module (DDM) listing of the file, you can complete the following to get additional metadata information:

Field	Action
Available Source Libraries	From the list of **Available Source Libraries**, select the virtual source library that contains the data structure definition that you want the virtual table to use.
Source Library Members	Select the names of each virtual source library member that represents the data structure that you want to include. The green arrow next to a DDM indicates that it is a suggested member, not that it is selected.

7. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Expand the source file to verify that it displays the expected data layout.
Start Field	This field is not supported for Adabas because the entire data layout is used.
End Field	This field is not supported for Adabas because the entire data layout is used.

8. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, “Generating and executing SQL queries,” on page 41.

Important: Use caution when using the BASE_KEY in WHERE predicates, (for example, `[PARENT TABLE].BASE_KEY = [CHILD TABLE].PARENT_KEY`) when joining the parent table with a child subtable, since this will result in a table scan of the entire Adabas file. It is recommended instead to use the CHILD_KEY (for example, `[PARENT TABLE].CHILD_KEY = [CHILD TABLE].PARENT_KEY`).

Creating virtual tables for RDBMS data sources

Create virtual tables that map to RDBMS data sources, such as Db2 for z/OS, Db2 LUW (Linux, UNIX, and Windows), Oracle, and Microsoft SQL Server.

About this task

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.

Db2 data access method:

When virtual tables are created for access to Db2 for z/OS data, an option is available to select the access method. *Db2 Direct* is a Data Service server access method that reads the data in the Db2 VSAM linear data sets directly instead of accessing the data through traditional Db2 APIs. For more information, see "Db2 for z/OS data access methods" in the *Solutions Guide*.

Note: The data access method options are not displayed if the Data Service server does not support Db2 Direct.

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.

Field	Action
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 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.
• **Use traditional DB2 access (read/write, transactional integrity)** • **Use DB2-Direct access (read-only, high performance bulk data access)**	Select the access method to use when accessing Db2 for z/OS data. Choose **Use traditional DB2 access (read/write, transactional integrity)** to use Db2 APIs such as DRDA, CAF, and RRSAF. This is the default selection. Choose **Use DB2-Direct access (read-only, high performance bulk data access)** to use Db2 Direct. **Note:** These options are available only when creating virtual tables for access to Db2 for z/OS data and if the Data Service server supports Db2 Direct.
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 Chapter 5, "Generating and executing SQL queries," on page 41.

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 Data Service 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. Virtual tables created for CA IDMS sets 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.
   - To select all child records under a parent, click the parent record.
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.
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 Data Service 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 Data Service 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 Chapter 5, “Generating and executing SQL queries,” on page 41.

Creating virtual tables for IMS data

Create a virtual table that maps to the IMS data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

The Program Specification Block (PSB) and Database Definition (DBD) source members, and the copybooks for each segment must exist in the virtual source libraries defined to the server. For details, see “Creating virtual source libraries ” on page 9.

To use the IMS Direct feature, the `IMSDIRECTENABLED` parameter must be enabled in the server IN00 file.

About this task

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 the database and file types supported as well as the size of the database.

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **IMS** wizard and click **Next**.
4. On the **New IMS virtual Table(s)** page, create metadata for an IMS virtual table by completing the following steps:
   a) Choose a DBD by doing one of the following steps:
      - Select a **DBD** from the drop-down list.
      - If your DBD does not appear in the drop-down list, click **Extract DBD** to create the requisite metadata. The **New IMS DBD Metadata Wizard** launches. See “Using the IMS DBD Metadata wizard” on page 17.

   b) Choose a PSB by doing one of the following steps:
      - Select a **PSB** from the drop-down list.
      - If your PSB does not appear in the drop-down list, click **Extract PSB** to create the requisite metadata. The **New IMS PSB Metadata Wizard** launches. See “Using the IMS PSB Metadata wizard” on page 18.

c) Click **Create Virtual Table** to create a virtual table for an IMS segment in the selected DBD and PSB. The **New Virtual Table Wizard** launches. See "Using the IMS Virtual Table wizard" on page 19.

   Note: Both the DBD and PSB must be defined for this button to be enabled.

5. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, "Generating and executing SQL queries," on page 41.

Using the IMS DBD Metadata wizard

Use the **New IMS DBD Metadata Wizard** to create DBD server metadata.

About this task

This wizard is used to create server metadata containing information extracted from the selected DBD source. This DBD metadata is a prerequisite for creating IMS virtual tables. The name of each DBD map will be determined from the contents of the DBD source.

Procedure

1. On the **New DBD Metadata** page, complete the following fields and click **Next**:

Field	Action
Metadata Library	From the drop-down list, select the target library where the DBD 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.

2. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of **Available Source Libraries**, select the virtual source library that contains the DBD source member.
Source Library Members	Select the DBD that you want to use and click **Download** to copy the member from the mainframe to your desktop. Use **Filter patterns** to filter the list.
Downloaded Source Files	Review the list of downloaded members and ensure that the check box for the DBD that you want to use has been selected.

3. On the **Data Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Expand the source file to verify that it displays the expected database definition (DBD).
Start Field	Accept the default root start field, or if multiple DBD nodes are present in the source tree, you can click on one of the DBD nodes to indicate that you only want to map that one DBD.
End Field	**End Field** selection is disabled when extracting DBD source.

4. On the **IMS Server configuration** page, complete the following fields:

Field	Action
• **Use IMS/DBCTL (read/write, transactional integrity)**	Select the IMS protocol to use. Choose **Use IMS/DBCTL (read/write, transactional integrity)** to use IMS API calls.

Field	Action
• **Use IMS-Direct (read-only, high performance bulk data access)**	Choose the default option **Use IMS-Direct (read-only, high performance bulk data access)** to enable IMS Direct for the DBD. To use this feature, IMS Direct must also be enabled in the server IN00 file. You must select this option for the DBD to be able to enable IMS Direct for a virtual table.
IMS ID Override (used with IMS-Direct only)	Specify the IMS ID of the IMS subsystem to use when multiple IMS subsystems are defined for use with IMS Direct. This value will override the default IMS ID in the DBD map.
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. Click **Finish**.

What to do next
Return to the **New IMS Virtual Table(s)** page and define the IMS PSB. See "Creating virtual tables for IMS data" on page 16.

Using the IMS PSB Metadata wizard

Use the **New IMS PSB Metadata Wizard** to create PSB server metadata.

About this task
This wizard is used to create server metadata containing information extracted from the selected PSB source. This PSB metadata is a prerequisite for creating IMS virtual tables. The name of each PSB map will be determined from the contents of the PSB source.

Procedure

1. On the **New PSB Metadata** page, complete the following fields and click **Next**:

Field	Action
Metadata Library	From the drop-down list, select the target library where the PSB 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.

2. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of **Available Source Libraries**, select the virtual source library that contains the PSB source member.
Source Library Members	Select the PSB that you want to use and click **Download** to copy the member from the mainframe to your desktop. Use **Filter patterns** to filter the list.
Downloaded Source Files	Review the list of downloaded members and ensure that the check box for the PSB that you want to use has been selected.

3. On the **Data Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Expand the source file to verify that it displays the expected program specification block (PSB).
Start Field	Accept the default root start field, or if multiple PSB nodes are present in the source tree, you can click on one of the PSB nodes to indicate that you only want to map that one PSB.
End Field	**End Field** selection is disabled when extracting DBD source.

4. Click **Finish**.

What to do next

Return to the **New IMS Virtual Table(s)** page and create the virtual table. See "Creating virtual tables for IMS data" on page 16.

Using the IMS Virtual Table wizard

Use the **New Virtual Table Wizard** to create a new IMS virtual table.

About this task

This wizard is used to map an IMS segment using a copybook representation to produce a new IMS virtual table.

Procedure

1. On the **New IMS Virtual Table** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
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.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
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.

2. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of **Available Source Libraries**, select the virtual source library that contains the data structure definition that you want the virtual table to use.
Source Library Members	Select the PDS members that represent the data structures to include and click **Download** to copy the members from the mainframe to your desktop.

Field	Action
Downloaded Source Files	Select one or more previously downloaded members.

3. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows: • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, **Enable End Field Selection** must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, **Enable End Field Selection** must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is not selected, and select a different start field in the **Source** tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the **Source** tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is selected, and select a different end field in the **Source** tree.

4. On the **IMS Information** page, complete the following fields:

Field	Action
Segment Name	From the drop-down list, select the segment name.
• **Use IMS/DBCTL (read/write, transactional integrity)** • **Use IMS-Direct (read-only, high performance bulk data access)**	Select the IMS protocol to use. Choose the default option **Use IMS/DBCTL (read/write, transactional integrity)** to use IMS API calls. Choose **Use IMS-Direct (read-only, high performance bulk data access)** to enable IMS Direct on the virtual table. To use this feature, IMS Direct must also be enabled for the selected DBD and enabled in the server IN00 file.
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. Click **Finish**.

What to do next

Return to the **New IMS Virtual Table(s)** page and if necessary create the next virtual table. See "Creating virtual tables for IMS data" on page 16.

Creating virtual tables for IBM MQ

Create a virtual table that maps to the IBM MQ data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

Before creating the virtual table, verify that the MQ queue exists and that the copybook exists in the source library. If you use delimited data, configure support for delimited data processing. See "Configuring delimited data support" in the *Installation and Customization Guide*.

About this task

Data in MQ queues is described using COBOL or PLI data descriptions taken from copybooks or programs.

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **MQ** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	**Action**
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
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.
Arrays Handling	Enable one of the following array management options: • **Flatten arrays into a single fixed table at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. • **Return arrays into separate tables at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. A subtable is generated for each array. Subtables only support SQL read access.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	**Action**
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click **Download** to copy the members from the mainframe to your desktop. Use **Filter patterns** to filter the list.
Downloaded Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	**Action**
Source	Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows:

Field	Action
	• For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, **Enable End Field Selection** must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, **Enable End Field Selection** must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is not selected, and select a different start field in the **Source** tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the **Source** tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is selected, and select a different end field in the **Source** tree.

7. On the **MQ Details** page, complete the following fields:

Field	Action
Queue Manager Name	Enter the IBM MQ queue manager name. The name is a four-character subsystem name.
Queue Name	Enter the IBM MQ queue name. The name can contain a maximum of 48 characters and must comply with z/OS data set naming standards.
Post-Read Exit Name	To manipulate the data after reading it from the queue, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.

8. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, “Generating and executing SQL queries,” on page 41.

Creating virtual tables for sequential data

Create a virtual table that maps to the sequential data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

Before creating the virtual table, verify that the data set name exists and that the copybook exists in the source library.

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **Sequential** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
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.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Enable one of the following array management options: • **Flatten arrays into a single fixed table at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. • **Return arrays into separate tables at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. A subtable is generated for each array. Subtables only support SQL read access. • **Flatten arrays now**: If you select this option, you cannot change array-handling after you save the virtual table.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click **Download** to copy the members from the mainframe to your desktop.
Download Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows: • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, **Enable End Field Selection** must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, **Enable End Field Selection** must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is not selected, and select a different start field in the **Source** tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the **Source** tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.

Field	Action
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is selected, and select a different end field in the **Source** tree.

7. Optional: On the **Virtual Table Redefines** page, accept the default table redefines or expand **Redefine** to modify your selection, and click **Next**.
8. On the **Data Source Details** page, complete the following data source fields and click **Next**:

Field	Action
Data Set Name	Enter the data set name you want to use. The following data set types are supported: • PDS or PDSE: Specify the partitioned data set name. This requires that you also enter a **Member** name prior to validating that the member name exists on the host. • Physical sequential: Specify the sequential data set name and click **Validate** to verify that the data set name exists on the host. • Generation Data Groups (GDG): Specify the GDG data set using the GDG syntax. For example: *hlq*.DATA.SEQ(-1). You can also specify a base GDG name so that all generations of the GDG will potentially be accessed. Click **Validate** to verify that the data set name exists on the host.
Member	If you selected a PDS or PDSE for the **Data Set Name**, you must also enter the member name to use. Click **Validate** to verify that the member name exists on the host.
Post-Read Exit Name	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.
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.

9. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, “Generating and executing SQL queries,” on page 41.

Creating virtual tables for VSAM, VSAM CICS, and IAM data

Create a virtual table that maps to the VSAM, VSAM CICS, and IAM data that you want to access, and from which the SQL used to access the data is generated and executed.

Before you begin

You must have the VSAM or VSAMCICS cluster name available (*sourcelibrary.copybook.filename*).

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **VSAM** wizard and click **Next**.
4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
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.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Enable one of the following array management options: • **Flatten arrays into a single fixed table at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. • **Return arrays into separate tables at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. A subtable is generated for each array. Subtables only support SQL read access. • **Flatten arrays now**: If you select this option, you cannot change array-handling after you save the virtual table.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Available Source Libraries	From the list of **Available Source Libraries**, select the virtual source library that contains the data structure definition that you want the virtual table to use.
Source Library Members	Select the PDS members that represent the data structures to include and click **Download** to copy the members from the mainframe to your desktop.
Download Source Files	Select one or more previously downloaded members.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows: • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, **Enable End Field Selection** must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, **Enable End Field Selection** must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is not selected, and select a different start field in the **Source** tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the **Source** tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.

Field	Action
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is selected, and select a different end field in the **Source** tree.

7. Optional: On the **Virtual Table Redefines** page, accept the default table redefines or expand **Redefine** to modify your selection, and click **Next**.
8. Complete the following VSAM related fields:

Field	Action
Cluster Name	Enter the cluster name for the VSAM data set, and click **Validate**. The server searches the catalog on the mainframe to confirm that the data set exists. If the data set exists, a dialog displays the data set type.
Post-Read Exit Name	To manipulate the data after reading it from the source file, enter the name of the post-read exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing after a record is read from the data source.
Pre-Write Exit Name	To manipulate the data before writing it to the source file, enter the name of the pre-exit to use. This is the custom exit routine that is installed on the server and is used to perform additional processing before a record is read from the data source.
Alternate Indexes	If the VSAM file has been defined to include alternate indexes, you can click **Get** to add index information to the virtual table, or you can click **Delete** to remove the information. Alternate indexes are used to improve query performance when the search criteria includes columns that are not part of the primary index. Alternate indexes have an indirect relationship to the cluster name, but they must be defined separately. If you are using a KSDS VSAM or ESDS cluster, you can specify alternative indexes that are associated with the cluster.
Advanced (VSAM only)	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.

9. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, “Generating and executing SQL queries,” on page 41.

Creating virtual tables for zFS and HFS file system data

Create a virtual table that maps to file data that you want to access on a zFS or HFS file system and from which the SQL used to access the data is generated and executed.

Before you begin

Before creating the virtual table, verify that the PDS members that represent the data structures for the data you want to virtualize already exist in the source library.

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. Right-click **Virtual Tables** and select **Create Virtual Table(s)**.
3. Under **Wizards**, select the **zFS** wizard and click **Next**.

4. On the **New Virtual Table Wizard** page, complete the following fields and click **Next**:

Field	Action
Name	Enter a unique name. The name can contain a maximum of 50 characters. The name must consist of an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character.
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.
Convert VAR* fields to True VAR* fields	This is a deprecated field and should not be selected.
Arrays Handling	Enable one of the following array management options: • **Flatten arrays into a single fixed table at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. • **Return arrays into separate tables at runtime**: This supports both **OCCURS** and **OCCURS DEPENDING ON** statements. A subtable is generated for each array. Subtables only support SQL read access.

5. On the **Source Download** page, complete the following fields and click **Next**:

Field	Action
Download Folder	Verify that the appropriate download folder is displayed.
Available Source Libraries	Select the source library that contains the data structure to use.
Source Library Members	Select the PDS members that represent the data structures to include and click **Download** to copy the members from the mainframe to your desktop.
Downloaded Source Files	Select one or more previously downloaded members. Selecting previously downloaded members is optional.

6. On the **Virtual Table Layout** page, complete the following fields and click **Next**:

Field	Action
Source	Browse the source tree to verify that it displays the expected data layout. By default, all of the fields in the tree will be included in the mapping. To include only a subset of the fields for the mapping, modify the start field value and, optionally, the end field value, as follows: • For the start field, accept the default root start field, or expand the tree and select a different start field. When selecting a different start field, **Enable End Field Selection** must not be selected. • For the end field, accept the default end field, or expand the tree and select a different end field. When selecting a different end field, **Enable End Field Selection** must be selected.
Start Field	Identifies the first field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is not selected, and select a different start field in the **Source** tree.
Enable End Field Selection	Use this field to control selection of the start field and end field values in the **Source** tree. When this option is not selected (default), you can select the start field. When this option is selected, you can select the end field.

Field	Action
End Field	Identifies the last field within the data layout that will be mapped. To change this value, make sure **Enable End Field Selection** is selected, and select a different end field in the **Source** tree.

7. On the **zFS Virtual Table Details** page, complete the following fields:

Field	Action
Pathname	Enter the path name of the zFS file. If the absolute path name of the zFS file is less than 255 characters in length, you must include the root slash "/" in the path name. For example, `/u/tsado/data/stuff.txt`. If the absolute path name of the zFS file is greater than 255 characters in length, you must enter the relative path name. The relative path name starts with the name of the target system to indicate the top-level directory and does not include the leading root slash. For example, `data/stuff.txt`, where "data" is the name of the target system.
Target System	If you plan to map several zFS files under the same zFS directory location, specify a target system to use. You can click **Create** to add a new path name to use, or if a relative path name is already specified in the **Pathname** field, you must select an existing target system from the drop-down list. If you choose to create a new target system, complete the following fields and click **Finish**: **Name** – Enter the name for the new target system. **CCSID** – Enter the CCSID of the character set in which the zFS file data is encoded. The default setting is **EBCDIC 1047**. **Base Pathname** – Enter the absolute path name under which the zFS file resides. Typically, this is the path name of the zFS subdirectory that contains your zFS file. At runtime, the server will determine the location of the zFS file by concatenating the path name with the value specified in the virtual table **Pathname** field. The server does not insert additional slash (/) separators when concatenating the target system path name and the virtual table path name. If the target system path name represents a complete directory name, include the trailing slash (`/tmp/`).
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.

8. Click **Finish**.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, “Generating and executing SQL queries,” on page 41.

Creating virtual tables for VSAM and sequential access using ADDI

Create virtual tables that map VSAM and sequential data for COBOL applications by using information made available through IBM Application Discovery and Delivery Intelligence (ADDI).

Before you begin

The Data Service server must be configured to access one or more ADDI projects hosted on Microsoft SQL Server. The studio recognizes ADDI when virtual views and target system maps are installed. Map recognition is based on target systems starting with the string TSIAD and virtual views starting with the name IADV_. For more information on configuring the server, see the *Solutions Guide*.

About this task

To create the virtual tables that are used to access VSAM and sequential data for COBOL applications, information is queried in the ADDI project. Information is retrieved about the z/OS data sets and the COBOL copybooks used to access the z/OS data sets.

The following restrictions and considerations apply:

- Virtual table creation is restricted to data sets in the ADDI project that are processed by COBOL programs using JCL. Data sets accessed using CICS as well as other databases (such as IMS, CA IDMS, or Adabas) are not supported.
- When retrieving data sets from the ADDI project, the studio provides a list of all data sets discovered in the ADDI project that correspond to copybook information. If the data set does not have a corresponding copybook, the data set will not be presented in the studio.
- When creating virtual tables in the studio, duplicate records may appear in the generated list. (Duplicate records have the same project and copybook record names but different ID values.) This is due to multiple copies of the same copybook existing in the ADDI project. The studio provides a feature that compares the definitions of the records and allows you to remove any duplicates.
- When mapping COBOL copybooks containing REDEFINES clauses, default mapping rules related to REDEFINES will be applied which will result in disabled columns in the maps. Editing of virtual maps may be required after generation to enable or disable generated columns.
- ADDI project names are limited to 13 characters due to location name restrictions in the z/OS server.

Procedure

1. On the **Server** tab, explore the ADDI metadata information by expanding the **Discovery** > **IBM Application Discovery** node, and then navigating down the appropriate subtree. The hierarchy begins with the project, followed by the data sets, and then the associated records.
2. Optional: Right-click a record and select **Display Data Layout** to show the copybook for the record.
3. Select one or more data sets or records to map, as follows:
   - To select individual data sets or records, hold down the Ctrl key and click each data set or record to include.
   - To select a range of data sets or records, click the first data set or record in the range, and then hold the Shift key and select the last data set or record in the range. All data sets or records within the range will be included.
   - To select all records under a data set, click the data set.
4. Right-click the selected data sets or records and select **Create Virtual Table(s)**.

   The **New Virtual Tables Wizard** launches, presenting a list of proposed virtual table names and the COBOL structure names that will be used as a basis to create columns for the virtual tables.
5. On the **Create virtual tables using IBM Application Discovery** 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.

Field	Description
Description	Enter an optional description.
Naming Pattern	Specify the format to use for the generated virtual table names. You can specify different patterns for the project name and records. Use the following variables to create naming patterns that are derived from the ADDI metadata: • {Project}: ADDI project name • {Record}: Record name
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. • **Flatten arrays now (C)**: If you select this option, you cannot change array-handling after you save the virtual table.

6. In the table that lists the records, review the list of selected entries and perform the following steps:
   a) Optional: If duplicate target virtual table names appear, which are identified with a description in the **Errors** column, click **Remove Duplicates**.
   The studio compares the definitions of the records and removes any duplicates.
   b) Click **Validate** to validate each data set and determine the data set type.

   The studio populates the **Type** column with the correct data set type.
   c) Modify the selections to map as needed.

   Tip: Use the check box in the header row of the table to control the selection of all entries.
7. Optional: Click **Advanced** to display and complete the following fields:

Field	Description
MapReduce (Server Parallelism Overrides)	When reading large volumes of data from tables, you can use 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.

8. Click **Finish**.

Results

The virtual tables are created on the server and are visible under the **SQL** > **Data** > ***SSID*** > **Virtual Tables** tree node, where *SSID* is the name of your server.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, “Generating and executing SQL queries,” on page 41.

Creating virtual tables for VSAM and sequential access using RAA

Create virtual tables that map VSAM and sequential data for COBOL applications by using information made available through IBM Rational Asset Analyzer (RAA).

Before you begin

The Data Service server must be configured to access one or more RAA database schemas hosted on DB2 for z/OS. The studio recognizes RAA when RAA virtual views and target system maps are installed. Map recognition is based on target systems starting with the string TSRAA and virtual views starting with the name RAAV_. For more information on configuring the server, see the *Solutions Guide*.

The preferred method to collect COBOL information is to retrieve record layouts directly from the WebSphere Application Server that hosts RAA. The WebSphere Application Server must be configured using the Metadata Discovery preferences. For more information, see “Metadata Discovery preferences” on page 59.

About this task

To create the virtual tables that are used to access VSAM and sequential data for COBOL applications, information is queried in the RAA database and from the host. Information is retrieved about the z/OS data sets and the COBOL copybooks used to access the z/OS data sets. If the WebSphere Application Server has been configured, all access to the host for record layout information will first be attempted using the WebSphere Application Server hosting RAA. If access to the RAA host fails and the record layout is stored in a PDS, layout retrieval will be attempted using the current Data Service server.

The following restrictions and considerations apply:

- Virtual table creation is restricted to data sets in the RAA database that are processed by COBOL programs using JCL. Data sets accessed using CICS as well as other databases (such as IMS, CA IDMS, or Adabas) are not supported.
- When retrieving data sets from the RAA database, the studio provides a list of all data sets discovered in the RAA database that correspond to copybook information. If the data set does not have a corresponding copybook, the data set will not be presented in the studio.
- When creating virtual tables in the studio, duplicate records may appear in the generated list. (Duplicate records have the same database and copybook record names but different ID values.) This is due to multiple copies of the same copybook existing in the RAA database. The studio provides a feature that compares the definitions of the records and allows you to remove any duplicates.
- When mapping COBOL copybooks containing REDEFINES clauses, default mapping rules related to REDEFINES will be applied which will result in disabled columns in the maps. Editing of virtual maps may be required after generation to enable or disable generated columns.

Procedure

1. On the **Server** tab, explore the RAA metadata information by expanding the **Discovery** > **IBM Rational Asset Analyzer** node, and then navigating down the appropriate subtree. The hierarchy begins with the database, followed by the data sets, and then the associated records.
2. Optional: Right-click a record and select **Display Data Layout** to show the copybook for the record.
3. Select one or more data sets or records to map, as follows:
   - To select individual data sets or records, hold down the Ctrl key and click each data set or record to include.
   - To select a range of data sets or records, click the first data set or record in the range, and then hold the Shift key and select the last data set or record in the range. All data sets or records within the range will be included.
   - To select all records under a data set, click the data set.
4. Right-click the selected data sets or records and select **Create Virtual Table(s)**.

   The **New Virtual Tables Wizard** launches, presenting a list of proposed virtual table names and the COBOL structure names that will be used as a basis to create columns for the virtual tables.

5. On the **Create virtual tables using IBM Rational Asset Analyzer** 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. You can specify different patterns for the database name and records. Use the following variables to create naming patterns that are derived from the RAA metadata: • {Database}: RAA database name • {Record}: Record name
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. • **Flatten arrays now (C)**: If you select this option, you cannot change array-handling after you save the virtual table.

6. In the table that lists the records, review the list of selected entries and perform the following steps:
   a) Optional: If duplicate target virtual table names appear, which are identified with a description in the **Errors** column, click **Remove Duplicates**.

      The studio compares the definitions of the records and removes any duplicates.

   b) Click **Validate** to validate the data set and determine the data set type.

      The studio populates the **Type** column with the correct data set type.

   c) Modify the selections to map as needed.

      Tip: Use the check box in the header row of the table to control the selection of all entries.
7. Optional: Click **Advanced** to display and complete the following fields:

Field	Description
MapReduce (Server Parallelism Overrides)	When reading large volumes of data from tables, you can use 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.

8. Click **Finish**.

Results

The virtual tables are created on the server and are visible under the **SQL** > **Data** > ***SSID*** > **Virtual Tables** tree node, where *SSID* is the name of your server.

What to do next

Use the studio to easily compose and execute SQL queries using your new virtual tables. See Chapter 5, "Generating and executing SQL queries," on page 41.

Creating virtual views

Consider creating a virtual view if columns in your virtual table are missing or if you want to join columns from different virtual tables.

Before you begin

The virtual tables representing the data that you want to access or join must already exist.

About this task

A virtual view comprises the SELECT statement that contains the columns from the source data that are used to read data directly from the data source. For example, `SELECT * FROM HLS_JOIN_VSAM LIMIT 1000;`. In some cases, creating virtual views is more convenient than regenerating and editing SQL each time you want to access the same data.

Procedure

1. In the **Server View**, expand **SQL** > **Data** > **Data Service server** > **Virtual Tables**.
2. Right-click the virtual table that represents the data that you want to access, and select **Create Virtual View**.
3. In the **Name** field, enter a name for the virtual view.
4. From the **Target** drop-down list, select the target to use for this virtual view.
5. Optional: In the **Description** field, enter a description.
6. Click **Next**.
7. In the **Table Browser**, expand the **Virtual Tables** folder, and select an existing virtual table to use to compose the SQL statement.
8. Click **Next**.
9. Optional: Review the resulting SQL statement and make any necessary modifications.
10. Click **Validate** to validate the SQL.
11. If valid, on the **SQL Validation** message that displays, click **OK**.
12. Click **Finish**.

Results

In the **Server** view, locate the new virtual view by expanding **SQL** > **Data** > **Data Service server** > **Virtual Views**.

Creating Db2 user-defined table functions

Use the **New UDTF Definitions in DB2 Wizard** to create user-defined table functions (UDTFs) in Db2 for z/OS for access to any supported data source type.

Before you begin

Db2 Virtualization (DB2V) is a feature that provides single-point access to various data source types. For additional information about configuring your system to use Db2 for z/OS as a primary access point, see "Using Db2 for z/OS to access multiple data source types" in the *Solutions Guide*.

To use this wizard, virtual tables should already exist for your data sources. See “Creating virtual tables” on page 10.

About this task

Use the **New UDTF Definitions in DB2 Wizard** to create the necessary user-defined table functions and views in Db2 for z/OS for access to any supported data source type. For existing virtual tables, this wizard creates the necessary objects in a local Db2 subsystem so that Data Service data can be queried using Db2 clients.

Procedure

1. Expand the **SQL** > **Data** > ***SSID*** node, where *SSID* is the name of your server.
2. In the **Virtual Tables** node, right-click one or more virtual tables, and select **Create UDTF Definitions in DB2**.
3. In the **New UDTF Definitions in DB2 Wizard**, on the **Generate DDL with user-defined table functions** page, complete the following fields:

Field	Action
General DB2 Settings	Specify information about the Db2 subsystem where the UDTFs and views will be created. • **Subsystem**: Select the Db2 subsystem ID from the drop-down list. • **Schema**: Select the schema from the drop-down list, or enter a new schema name. • **GRANT TO**: Specify to whom privileges are granted for the generated UDTFs and views. Clear this field to not include the GRANT TO statement in the generated DDL. • **UDTF Module**: This value defaults to the UDTF module in use for your system. If you need to change this value, enter the name of another UDTF module. The following modules are available: – AZKUDT9N – AZKUDTAN – AZKUDTBN – AZKUDTCN For more information, see "Using Db2 for z/OS to access multiple data source types" in the *Solutions Guide*. • **WLM Environment**: The address space Db2 starts to run user-defined functions. Leave this field blank to omit the WLM ENVIRONMENT clause in the generated DDL.
"Generate" Actions	Specify whether to execute or save the DDL, or both. • **Execute generated DDL in DB2**: Select this option to execute the generated DDL on the specified Db2 subsystem. • **Save DDL to file**: Optionally, enter a file name (with `.sql` extension) where to save the DDL. This step might be required if you do not have authorization to execute the DDL. Or, you might want to review the DDL in the **SQL Editor** first, before running it in Db2. – **Append to file**: Select to append the generated DDL to an existing file. – **Open file in SQL Editor**: Select to open the generated DDL in the **SQL Editor**.
Naming Patterns	Specify the format to use for the generated function and view names. Use the following variables to create naming patterns: • {Server}: Data Service server name • {Table}: Virtual table name The **Function Name** and **View Name** columns are editable in the table, so you can also customize the names on an individual basis. **Note:** If **Views** is blank, views will not be generated.

4. Click **Generate**.

Results

The DDL for creating UDTFs and corresponding views is generated. For more information, see "Generated DDL for UDTFs" on page 35.

After the DDL is executed, a UDTF and a corresponding view are created for each selected virtual table. In the **Server** tab, locate the new objects by expanding **SQL** > **Data** > **Other Subsystems** > ***db2SSID*** > ***schema***, and then the appropriate nodes, as follows:

- The Db2 views are located in the **Views** node.
- The Db2 UDTFs are located in the **DB2V** > **User-Defined Table Functions** node.

What to do next

After the Db2 views and UDTFs have been created, you can perform the following tasks:

- Using the new Db2 views, compose and execute SQL queries to access the data. You can do this from the Data Service studio or from a Db2 client.
- Using the new Db2 UDTFs, compose and execute SQL queries to select data from the virtual table. Queries can be generated for a specific UDTF function or for a subset of columns within a UDTF.

See Chapter 5, "Generating and executing SQL queries," on page 41.

Generated DDL for UDTFs

This topic describes the DDL that is generated by the **New UDTF Definitions in DB2 Wizard** in the Data Service studio when creating Db2 UDTFs and corresponding views for virtual tables.

For each Data Service map created as a view in Db2, there are two DDL statements defining catalog objects to Db2. The first statement is a CREATE FUNCTION statement, which describes the user-defined function to retrieve data from Data Service. The second statement is a CREATE VIEW statement, which calls the user-defined table function.

Example: CREATE FUNCTION

The following example shows the Db2 DDL generated to define the user-defined table function for the STAFFVS table:

```
CREATE FUNCTION "TSUSER"."AZKS_STAFFVS"
 (CONDITION VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,
  DVMNAME VARCHAR(254) CCSID EBCDIC FOR SBCS DATA,
  COLINFO  VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,
  REQUEST VARCHAR(254) CCSID EBCDIC FOR SBCS DATA)
 RETURNS TABLE
  ("STAFFVS_KEY_ID" SMALLINT,
   "STAFFVS_DATA_NAME_L" SMALLINT,
   "STAFFVS_DATA_NAME" CHAR(9),
   "STAFFVS_DATA_DEPT" SMALLINT,
   "STAFFVS_DATA_JOB" CHAR(5),
   "STAFFVS_DATA_YRS" SMALLINT,
   "STAFFVS_DATA_FILLER" CHAR(8))
 EXTERNAL NAME "AZKUDTCN"
 LANGUAGE ASSEMBLE
 PARAMETER STYLE DB2SQL
 DETERMINISTIC
 FENCED
 NO SQL
 SCRATCHPAD 2048
 FINAL CALL
 DISALLOW PARALLEL
 DBINFO WLM ENVIRONMENT DSN1WLM1
 STAY RESIDENT YES
 ASUTIME NO LIMIT
 SECURITY USER;
```

Note the following about the UDTF DDL:

- The following parameters are common to the generic table function:

```
(CONDITION VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,
 DVMNAME VARCHAR(254) CCSID EBCDIC FOR SBCS DATA,
 COLINFO  VARCHAR(24576) CCSID EBCDIC FOR SBCS DATA,
 REQUEST VARCHAR(254) CCSID EBCDIC FOR SBCS DATA)
```

  These parameters are required on every UDTF definition for a Data Service virtual table referenced by a view in Db2. For more information about these parameters, see "UDTF parameters" on page 36.
- The `RETURNS TABLE` clause describes the columns in the Data Service table as defined in the map.

  Warning: These columns must match exactly the map definition to prevent errors as the UDTF does not have access to this information at invocation time and therefore does no SQL type conversion.

  Note that the system parameter SQLENGCHARTOVARCHAR changes column definitions and must be considered when defining the UDTF. Because the `RETURNS TABLE` clause describes the return table data to Db2, each map requires a separate UDTF definition. While tables having identical columns can technically share the same definition, this practice is not recommended unless the table is from a common map shared by multiple Data Service servers.
- The following DDL elements contain information you can customize at view creation time. Considerations for this information are as follows:
  - `CREATE FUNCTION "TSUSER"."AZKS_STAFFVS"` – Like most DB2 catalog objects, function names include both a schema name and a function name. The studio wizard provides flexibility in both the schema and function naming, including a default pattern for function names.
  - `EXTERNAL NAME "AZKUDTCN"` – The external name will always be in the form AZKUDT*α*N, where *α* is the z/OS architecture level (9, A, B, C).
  - `WLM ENVIRONMENT DSN1WLM1` – Identifies the name of the address space that Db2 starts for running user-defined functions. A reasonable default is the Db2 subsystem ID suffixed with WLM1, which the user can change if necessary. This element is optional, and when omitted, the default WLM environment for that Db2 subsystem will be chosen at runtime.

Example: CREATE VIEW

The following example shows the Db2 DDL generated to define the view to call the user-defined table function:

```
CREATE VIEW "TSUSER"."AZKS_VSTAFFVS" AS
 SELECT
"STAFFVS_KEY_ID","STAFFVS_DATA_NAME_L","STAFFVS_DATA_NAME","STAFFVS_DATA_DEPT",
  "STAFFVS_DATA_JOB","STAFFVS_DATA_YRS","STAFFVS_DATA_FILLER"
  FROM TABLE("TSUSER"."AZKS_STAFFVS"
 ('',
  'AZKS...STAFFVS',
  '7,STAFFVS_KEY_ID,STAFFVS_DATA_NAME_L,STAFFVS_DATA_NAME,STAFFVS_DATA_DEPT,' ||
   'STAFFVS_DATA_JOB,STAFFVS_DATA_YRS,STAFFVS_DATA_FILLER',
  '')
  CARDINALITY 10000);
```

The CARDINALITY clause value is controlled by the **DB2 CARDINALITY in generated UDTF query** setting in "SQL preferences" on page 59.

UDTF parameters

The following table describes the parameters that must be passed when calling the UDTF.

Parameter	Description
`CONDITION ('')`	The CONDITION parameter can be used to add a WHERE predicate to the generated SQL sent to the Data Service server. For example, if you want to create a Db2 view that returns all managers using the STAFFVS example, you could specify `'WHERE STAFFVS_DATA_JOB = ''MGR'''` in the CONDITION parameter. Generally, this value will be defined as `''`. **Note:** By using the WHERE predicate on the UDTF call in the CONDITION parameter, the result set is filtered on the UDTF call. If you use the WHERE predicate when querying the view instead of passing it as a CONDITION parameter, all results will be returned on the UDTF call first and then filtered on the view call. This might affect performance.
`DVMNAME ('AZKS...STAFFVS')`	The DVMNAME is a four-part period-separated name in the form *dddd.bbbb.ssssssss.vvvvvvvv* where: • *dddd* – The 4-character subsystem name or 5-8 character group name for the local Data Service server. If the name is greater than 4 characters, the token is assumed to be a group name. • *bbbb* – The 4-character subsystem name if the table is in another Db2 subsystem. Generally, this token will be omitted. • *ssssssss* – The schema name for the table in the Data Service server. This token is for future use and should be omitted. • *vvvvvvvv* – The virtual table name in the Data Service server.
`COLINFO ('7,STAFFVS_KEY_ID,...')`	Describes the column count and optionally the Data Service virtual table column name list to the table function. Minimally, this must include the number of columns in the return table (for example, `'7'`). The form shown in the example includes a comma-separated list of every column in the table in COLNO order. If provided, this list will be used to generate optimized queries when the Db2 user queries a subset of columns in the view definition. **Warning:** It is critical that this count matches the number of columns included in the RETURNS TABLE clause of the CREATE FUNCTION DDL.
`REQUEST ('')`	Specifies runtime options. See the next section, "REQUEST runtime options" on page 38

REQUEST runtime options

Runtime options can be passed as comma-separated values in the REQUEST parameter. In most cases, these values are used to diagnose problems with the UDTF and should be added under the direction of IBM Software Support. The following runtime options can be added to the REQUEST parameter to control execution of the table function:

Option	Description
GROUPNAME	Instructs the table function to use the first part of DVMNAME as a Data Service server group name instead of a Data Service subsystem ID. It is only needed if the first token is 4 or less characters in length, but can also be included for documentation purposes if desired (for example, 'GROUPNAME').
MRC(n)	Enables MRC or MRCC processing. When specified without MRID, the UDTF will create multiple MRC connections to the Data Service server and partition row data across connections based on the map/reduce partitioning algorithm for the underlying target database or data set. When specified with MRID, MRC will act as a single participant in an MRCC request (for example, 'MRC(6),MRID(1)').
MRID(n)	Used in conjunction with MRC to identify the map/reduce participant ID associated with a view in MRCC request environments. Db2 views set up with MRC and MRID will read a subset of the virtual table data based on the map/reduce partitioning algorithm for the underlying target database or data set (for example, 'MRC(6),MRID(1)')
TRACE()	Instructs the UDTF to send trace information to the Data Service server after a successful connection is open to the server. In the server trace, all trace information is enclosed between the XML tokens `<DVUDFT_TRACE>` and `</DVUDFT_TRACE>`. The following comma-separated sub-options can be specified within the TRACE() option. • BUILDINFO – Displays the build date and architecture level of the UDTF program. This trace is issued immediately after a successful connection is established to the Data Service server (for example, ‘TRACE(BUILDINFO)’). • CLOSE – Displays a message when a UDTF query is closed (for example, ‘TRACE(CLOSE)’). • SQL – Displays the generated SQL sent to the Data Service server in response to the UDTF call from Db2 (for example, ‘TRACE(SQL)’). • STATS – Displays a summary of the statistics at query close time, including rows retrieved and producer/consumer wait times (for example, ‘TRACE(STATS)’).
VPIO(n)	When used with VPDNAME, VPIO sets the number of I/O threads for a VPD group (for example, ‘VPNAME(VPG1),VPNO(6),VPIO(3)’).
VPNAME(name)	Enables VPD processing by defining a VPD group name (for example, ‘VPNAME(VPG1),VPNO(6),VPIO(3)’).
VPNO(n)	Sets the number of members in a VPD group (for example, ‘VPNAME(VPG1),VPNO(6),VPIO(3)’).
VPTO(n)	Sets the timeout value in seconds for a VPD group (for example, ‘VPNAME(VPG1),VPNO(6),VPIO(3),VPTO(10)’).

UDTF generated query example

The following example shows a query of a Db2 UDTF, generated in the Data Service studio, where a subset of the virtual table columns have been selected:

```
-- Description:   Retrieve the result set for AZKS_STAFFVS
-- Tree Location: rs99/46000/SQL/Data/Other Subsystems/DSN1/TSUSER/DB2V/
User-Defined Table Functions/AZKS_STAFFVS
-- Remarks:       DB2V:AZKS...STAFFVS
SELECT "STAFFVS_DATA_NAME","STAFFVS_DATA_DEPT","STAFFVS_DATA_JOB","STAFFVS_DATA_YRS"
   FROM TABLE("TSUSER"."AZKS_STAFFVS"
  ('',
   'AZKS...STAFFVS',
   '7,STAFFVS_KEY_ID,STAFFVS_DATA_NAME_L,STAFFVS_DATA_NAME,STAFFVS_DATA_DEPT,' ||
    'STAFFVS_DATA_JOB,STAFFVS_DATA_YRS',
   '')
   CARDINALITY 10000);
```

You can then modify the UDTF arguments in the generated SQL, such as the following options:

- Use the CONDITION parameter to add a WHERE predicate
- Use the TRACE parameter to send trace information to the server

The following example shows these modifications:

```
SELECT "STAFFVS_DATA_NAME","STAFFVS_DATA_DEPT","STAFFVS_DATA_JOB","STAFFVS_DATA_YRS"
   FROM TABLE("TSUSER"."AZKS_STAFFVS"
  ('WHERE STAFFVS_DATA_YRS > 5',
   'AZKS...STAFFVS',
   '7,STAFFVS_KEY_ID,STAFFVS_DATA_NAME_L,STAFFVS_DATA_NAME,STAFFVS_DATA_DEPT,' ||
    'STAFFVS_DATA_JOB,STAFFVS_DATA_YRS',
   'TRACE(BUILDINFO,SQL,STATS)')
   CARDINALITY 10000);
```

Chapter 5. Generating and executing SQL queries

To test SQL access to your data, generate and execute a SQL query from an existing virtual table or virtual view.

Before you begin

To avoid fetching large result sets that are memory intensive, the Data Service Studio provides settings related to SQL generation and retrieval that can limit the amount of data that is actually retrieved for a particular query execution. For more information, see "SQL preferences" on page 59.

Important: When writing SQL to access Adabas data, use caution when using the BASE_KEY in WHERE predicates, (for example, `[PARENT TABLE].BASE_KEY = [CHILD TABLE].PARENT_KEY`) when joining the parent table with a child subtable, since this will result in a table scan of the entire Adabas file. It is recommended instead to use the CHILD_KEY (for example, `[PARENT TABLE].CHILD_KEY = [CHILD TABLE].PARENT_KEY`).

Procedure

1. On the **Server** tab, right-click a virtual table and select **Generate Query**.
2. Choose from the following options:
   - **Execute –** Generate the SQL query in the **Data Source Editor** and execute the query.
   - **Cancel –** Generate the SQL query in the **SQL Editor** without executing the query. The generated SQL SELECT statement has all columns from the selected table. If the table contains a large number of columns, to avoid enumerating the various column names you can choose all columns using the **Generate Query with *** option.
3. Optional: In the **SQL Editor** view, modify the SQL to select only the data that you want to access. Any ANSI compliant SQL is acceptable.
4. To view or test the data that the SQL statement returns, right-click the highlighted SELECT statement and click either **Execute SQL** to view results in the **SQL Results** view, or **Execute SQL and File results** to save the results in a `.csv` file.
5. Optional: To create a virtual view of the SQL, highlight the SELECT statement, right-click and select **Create a virtual view**.

Results

In the **SQL Results** view:

- Double-click a row to view additional details about that row.
- Select the **Export Result Set** view option to export the SQL results to a `.csv` file.
- Click **SQL Messages** to view query-related system messages.

By default, if a result set includes 25 or more columns, each set of 25 columns are displayed incrementally as groups. You can choose which group you want to view using the **Columns Group** field. You can set the number of columns that you want to include in each group, ranging from 25-200, in the **Columns per group** field.

To change how SQL results display in the **SQL Results** view, see "Data Service preferences" on page 57.

What to do next

After the SQL statement is generated, you can perform any of the following tasks:

- Modify the SQL to meet your needs
- Execute the SQL to test and view the resulting data
- Create virtual views to join data or include missing columns

- Generate a SQL class to get access to data from your programs or applications

Chapter 6. Generating code from SQL

Use the **Code** wizard to generate the code that is used to get SQL access to data from your programs or applications.

Before you begin

The virtual table or virtual view that maps to the data that you want to access must already exist on the server.

Procedure

1. To launch the **Code** wizard from the **Server** tab, right-click the virtual table or virtual view and select **Generate Code From SQL**.

   Alternatively, to compose and execute your SQL query directly from a selected SQL statement in the editor, right-click on the selected SQL statement and select **Generate Code From SQL**.
2. On the **SQL** page, accept or change the file name that will be used to store the generated code.
3. Select from the following query options and click **Next**:
   - **Use Selected View –** Creates a simple query. This is the default setting.
   - **Compose the SQL Statement –** Selects all table columns and displays the resulting SQL statement. If necessary, you can choose to modify the resulting SQL statement before continuing to the next step.
4. On the **Code Generation** page, choose the programming language that you want to use to generate the SQL:
   - **Java Class**
   - **Java Spark Application**
   - **Scala Jupyter Notebook**

     Note: The generated Scala Jupyter Notebook code provides a simple starter program showing you how to use the JDBC driver to load data into a Spark application. The wizard does not allow SQL parameter markers for this application type.
   - **Scala Spark Application**
5. To finish generating the code, complete one of the following options:
   - If you did not choose the **Scala Jupyter Notebook** option, click **Finish**.
   - If you did chose the **Scala Jupyter Notebook** option, complete the following fields and click **Finish**:

Field	**Action**
Jupyter Kernel Name	Enter the name of the kernel to use.
Include additional Jars in the Generated Code	Select to include additional JAR files when generating the code. This setting is optional and is only available after you set the Generate Code preferences to include additional Jar files.
Credentials processing in generated code	Select from the following credential processing options: • **Omit password text –** the generated code will contain *** for the password variable, and will need to be updated manually. • **Use password text –** the password is included in the generated code in hashed format. • **Use INI file –** if this option is chosen, you must also specify the INI data set name to indicate that the DSN section in the INI file contains the user and

Field	Action
	password settings. Click **Sample** to generate and reference a sample INI file on your local system. This file must be available on the host where Jupyter Spark is running.
Preferences	Click **Preferences** to review the default settings for code generation.

Results

The generated code is saved in your workspace and is accessible from the Data Service Studio **Client** tab. The resulting file opens and can be modified in the editor if the **Scala Jupyter Notebook** option was not selected. Otherwise, the resulting file can be uploaded to Jupyter using the Jupyter Web UI.

Chapter 7. Accessing IT Operational Analytics data

To access, analyze, and report IT Operational Analytics (ITOA) data, generate the SQL from ITOA virtual tables.

When you configure the server, you have the option to include pre-defined data maps that administrators can use to access the following types of ITOA data:

- IBM System Management Facilities files (SMF)
- Operations Log files (OPERLOG_SYSLOG)
- System Log files (SYSLOG)

After you have configured the Data Service server to use ITOA pre-defined data maps, you can generate the SQL that is used to access ITOA data from the ITOA virtual tables.

For information about configuring access to operational analytics data with pre-defined data maps, see "Configuring access to SMF data for IT Operational Analytics" in the *Solutions Guide*.

System Management File sample code

Use SMF virtual tables to get SQL access to data in System Management Files (SMF).

About this task

When accessing data in SMF files, you use predefined virtual columns that are defined in the SMF virtual table map.

When using SMF log streams, you can use the following virtual columns to retrieve timestamp values:

LS_TIMESTAMP
Timestamp for log stream in GMT. When used in a WHERE predicate, the timestamp is searched in GMT.

LS_TIMESTAMP_LOCAL
Timestamp for log stream in local time zone. When used in a WHERE predicate, the timestamp is searched as local time.

To get SQL access to SMF data, complete the procedure that follows.

Procedure

1. From the Server view, expand **SQL** > **Data** > **server name** > **Virtual Tables**.
2. Right-click the SMF virtual table or view from which you want to access the data.
3. Right-click **Generate Query**, and then review the resulting SQL statement. If necessary, you can modify the statement to meet your needs. The following shows a sample SQL statement:

```
-- ------------------------------------------------------------------ Name         : SMF_00000
-- This statement will return all rows and all columns from the
-- following table:
-- Name         : SMF_00000 : null
-- Catalog      : null
-- Schema       : DVSQL
-- Remarks      : DATA - SMFDATA
-- Tree Location: rs28/1200/SQL/Data/VDBS/Virtual Tables/SMF_00000
-- The sql statement:
SELECT SMF_LEN, SMF_ZERO, SMF_FLAG, SMF_RTY, SMF_TIME, SMF_SID, SMF_SSI,
  SMF_STY, SMF_SEQN, SMF0JWT, SMF0BUF, SMF0VST, SMF0OPT, SMF0RST, SMF0RSV,
  SMF0OSL, SMF0SYN, SMF0SYP, SMF0TZ, SMF0MSWT, SMF0MTWT
FROM SMF_00000 LIMIT 1000;
```

4. Optional: Execute the SQL statement to view, test, or save the resulting data.

What to do next

Get the code to use in your programs and applications by creating a SQL class from the virtual table.

Chapter 8. Accessing DB2 unload data

Using existing DB2 virtual table definitions, you can issue SQL queries against your DB2 sequential unload data sets.

Before you can access your DB2 unload data using your DB2 virtual tables, you must configure access to the DB2 sequential unload data set. This access is configured using a virtual table rule. VTB rule AZKMDLDU is provided to demonstrate redirecting a DB2 virtual table to a DB2 unload data set. For information about setting up access, see "Configuring access to DB2 unload data sets" in the *Installation and Customization Guide*.

After you have performed the configuration steps, you can generate the SQL that is used to access the DB2 unload data using your existing DB2 virtual tables.

As an example, consider a virtual table named DSNA_EMPLOYEES that maps the EMPLOYEES table in DB2 subsystem DSNA. With the virtual table rule that specifies the DB2 unload data set enabled, you can query an unload sequential dataset named EMPLOYEE.UNLOAD.SEQ by issuing the following query:

```
SELECT * FROM MDLDU_DSNA_EMPLOYEES__EMPLOYEE_UNLOAD_SEQ
```

The rule performs the necessary steps to access the unload data set directly.

The following restrictions and considerations apply when using this feature:

- SQL access to DB2 unload files is limited to SQL queries only.
- The columns in the DB2 virtual table definition must exactly match the table unloaded in DB2.

Chapter 9. Server Trace

Use the **Server Trace** view to record and view Data Service server messages.

In the **Server Trace** view, you can perform the following tasks:

To collect and view diagnostics for the client, run the **Gather Diagnostics** wizard, which saves the information to a `.zip` folder.

Enabling Data Service Studio calls in the Server Trace results

To include Data Service Studio trace calls in your Server Trace results, enable the Data Service **Enable Server Tracing of Studio Calls** preference.

Before you begin

You must be able to connect to the Data Service server from which you want to collect trace information.

Procedure

1. From the **Window** menu, select **Open Preferences** > **Data Service**.
2. To enable tracing, select the **Enable Server Tracing of Studio Calls** check box. **Enable Server Tracing of Studio Calls** is enabled by default.
3. In the Data Service Studio **HTTP Debug Option** drop-down list, select one of the following HTTP debug options:

Field	Action
Off	Do not collect HTTP messages. All trace activities are deactivated, including interactive tracing.
Normal	Commands that complete with a failure status are traced after execution, including the return codes.
All	All instructions are traced before execution.
Commands	All commands are traced before execution. Return codes are also traced for commands that complete with an error or failure status.
Error	Commands that complete with error status are traced after execution, including the return codes.
Failure	Commands that complete with a failure status are traced after execution, including the return codes.
Intermediates	All instructions are traced before execution. All terms, intermediate results, and substituted variable names are traced during expression evaluation. The final results of any expression that is evaluated also displays. Values assigned by `arg`, `parse`, or `pull` instructions are also traced.
Labels	Shows all labels when executed.
Results	All instructions are traced before execution. The final result of any expression that is evaluated also displays. Values assigned by `arg`, `parse`, or `pull` instructions are also traced.

Starting Server Trace

Start tracing Data Service server records in the **Server Trace** view.

Before you begin

Before running **Server Trace**, you must be able to connect to the Data Service server from which you want to collect the trace information.

Procedure

1. From the **Studio Navigator** view, on the **Common Tools** tab, click **Server Trace**.
2. To start tracing, click **Play** (the blue arrow).
   The **Server Trace** table displays trace records.
3. Optional: To view message details, double-click the message and the details are displayed on the **Server Trace Zoom** page.
   You can also choose to search for specific details within the message.

Filtering Server Trace results

Use the **Profile** option to filter the records that display in the **Server Trace** view.

Before you begin

You must be able to connect to the Data Service server from which you want to filter trace information. You can set filtering criteria before or after you run a Server Trace. Your most current filtering selections are automatically saved as your default filtering profile.

Procedure

1. On the **Server Trace** view, click **Profile**.
2. On the **Server Trace Profile** page, enable the fields that you want to include in the results.
3. For each enabled field, click **Add** to further filter your results. You can either select from the values that are displayed or enter the value when prompted.
4. Click **OK** to save changes to your profile and to apply the profile to the results in the **Server Trace** table.

What to do next

Use the **Display** option to select and sort columns that display in the filtered table. You can also choose to export the trace results.

Using Server Trace Zoom

Use **Server Trace Zoom** view to view Server Trace message details.

Before you begin

Server Trace must be running before you can open the **Server Trace Zoom** view.

Procedure

1. In the **Server Trace** view, double-click the message for which you want to view details.
2. In the **Server Trace Zoom** view, view message details and choose from the following options:

Field	Action
Previous	Click **Previous** to search for the previous occurrence of the text string entered.
Next	Click **Next** to search for the next occurrence of the text string entered.
Search	Click **Search** and enter a search string. To search for the next occurrence of the text string, click **Search** again.
Close	Click **Close** to close the search dialog.

Searching Server Trace messages

You can search Server Trace message results for a particular text string or message ID.

Before you begin

You must start the Server Trace before you can begin searching within the resulting Server Trace messages.

Procedure

1. On the **Server Trace** view, click the drop-down menu, and select **Search**.
2. On the **Search** page that is displayed, in the **From** section, select one the following options to specify how to search within the results:

Field	Action
First	Search for the first occurrence of the text string.
Last	Search for the last occurrence of the text string.
ID	Search starting from the message ID you enter.

3. In the **For** field, enter the text string to use for searching within the message control blocks. Text strings cannot include spaces or special characters, and wild card searches are not supported.
4. Select **Previous** to find previous occurrences of the text string, or select **Next** to find the next occurrence of the text string.
5. Click **Search** to begin the search.

What to do next

View messages that meet the search criteria in the **Server Trace** view.

Labeling Server Trace messages

Create labels to bookmark server trace messages that you frequently access.

Before you begin

You must start the Server Trace before you can begin labeling messages.

Procedure

1. In the **Server Trace** view, right-click the message that you want to label and select **Add Label**.
2. On the **Message Label** dialog, enter text for the **Label** and click **OK**.
3. Optional: In the **Labels** view, double-click the label to locate the message in the **Server Trace** view.

Exporting Server Trace messages

Use the **Server Trace** view to export server trace messages as either ISX or CVS files.

About this task

You can limit the number of messages that you can export into a file by setting the **Server Trace export size limit** on the **Admin** preferences page.

Procedure

1. In the **Server Trace** view, from the drop-down menu, select **Export**.
2. Under **Export Type**, select one of the following message export options:

Field	**Action**
Summary	Exports the following minimum message information: • Message ID • Date • Time • User ID • Message text
Full	Exports all available message information and all data about that message including: • Message ID • Date • Time • User ID • Message text • Zoom
Comma Separated Format	Exports all table information to a CVS file. This file type cannot be imported for viewing in the **Server Trace** view.

3. Under **Export Content**, select one of the following message content options:

Field	**Action**
Message ID Range	Select a range of messages to export by entering the first message ID in **From**, and the last message ID to include in **To**.
Transaction ID	Exports only those messages with the RRS transaction ID value that you specify.
Global Transaction ID	Exports only those messages with the RRS global transaction ID that you specify.
Connection ID	Exports only those messages that are associated with a specific client that is currently connected to the server.
Message ID List	Lists message IDs. This option is only available if the **Full** export type option is selected.

4. Click **Next**.
5. On the **Export File** page, click **Browse** to specify a file name and export location.
6. Click **Finish**.

Importing Server Trace messages

To import and view Server Trace messages, use the **Import File Viewer** tab.

Before you begin

Server Trace must be running before you can import a file.

Procedure

1. In the **Server Trace** view, click the **Import File Viewer** tab and click **Import**.
2. Navigate to the ISX file that you want to import.
3. Double-click the ISX file. Messages and message details display on the **Import File Viewer** tab.
4. Optional: To view more details about a message, right-click on the message and select **Zoom**.
5. Optional: To change how the messages display, click **Display**.

Chapter 10. Data Service preferences

Preferences allow you to customize several Mainframe Data Service settings.

To view preferences, from the **Window** menu, select **Open Preferences** > **Data Service**.

Admin preferences

Use **Admin** preferences to set the maximum number of Server Trace messages that you want to export and to enable the tracing of DS Studio calls in the **Server Trace** view.

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

Field	**Description**
Server Trace export size limit	Sets the maximum number of messages to export. The default value is 5000. Specifying a value greater than 5000 can cause a MAX CPU TIME EXCEEDED error to occur.
Enable Server Tracing of Studio Calls	Includes Data Service Studio trace calls in your Server Trace results. This setting is disabled by default.

Code generation preferences

Use **Code Generation** preferences to customize how your code is generated.

Code Generation preferences are identified and described in the table that follows.

Field	**Description**
Jupyter Kernel Name	The name of the kernel to use as defined in your Jupyter configuration.
Jupyter Kernel Name	Identifies any additional JAR files that are not yet defined in the Spark configuration and that are required at runtime. The URL, `file://`, or `http://` naming format must be used. For example, you can add the following required JDBC driver JAR files: • file:///opt/dv-jdbc/dv-jdbc-3.1.201608041929.jar • file:///opt/dv-jdbc/log4j-api-2.6.2.jar • file:///opt/dv-jdbc/log4j-core-2.6.2.jar
INI Filename (as credentials store)	The INI file name that serves as your credentials store. The default setting is blank ( ).
INI Data Set Name (DSN)	The INI file name to use for DSN. The INI file name that you choose to enter can contain multiple credentials. The DSN name is used to identify each set of credentials. For example, you could use the mainframe userid in the DSN name. The default setting is blank ( ).

Console preferences

Use **Console** preferences to view or modify console display settings.

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

Field	**Description**
Fixed width console	Enable to specify a maximum number of characters to display in the console. This setting is disabled by default. **Maximum character width**: If **Fixed width console** is enabled, enter the maximum number of characters to display in the console. The default is 80 characters.
Limit console output	Enable to limit the console buffer and entry sizes by setting the maximum number of characters permitted: • **Console buffer size (characters)**. The default setting is 80000. • **Console entry size limit (characters)**. The default setting is 500.

Dictionary preferences

Use **Dictionary** preferences to add or delete reserved words in dictionaries, and add or delete dictionaries based on the languages being used.

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

Field	**Description**
Dictionary	Lists the default dictionaries. You can add new dictionaries to the list or delete existing dictionaries from the list.
Reserved word	Lists reserved words for each dictionary. You can add new words to the list or delete existing words from the list.

Driver preferences

Use **Driver** preferences to enable JDBC driver tracing and to specify the default location of the driver configuration files.

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

Field	**Description**
Enable Tracing	Enables tracing for the JDBC driver. If you change this option, you must restart the Data Service Studio to complete the change. This setting is disabled by default. **Note:** You can also access data sources that are stored in other configuration files, by adding those configuration files from the **Client** view.
Default DSN Config File	Specifies the default location of the DSN file. This file is used to store the JDBC connection definitions that are generated for use in the **Active Connections** view.

Field	Description
Connection Overrides	To override the connection settings that the Data Service Studio uses when it creates JDBC connection definitions, specify a single name-value pair or a semicolon-delimited list to be used. The default setting is a blank field ( ).

Data Service preferences

Use **Data Service** preferences to set preferences such as general session and SQL results settings.

General **Data Service** preferences are identified and described in the table that follows.

Field	Description
Enable Server Tracing of Studio Calls	Includes the Data Service Studio trace calls in your server trace results. This setting is disabled by default.
Studio HTTP Debug Option	The Data Service 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 Data Service 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 Data Service Studio operation is unsuccessful. The default setting is **30**.
Studio Remote Control Port	The port number that the Data Service 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*.`SAZKEXEC(AZKSIN00)` file: `"MODIFY PARM NAME(PASSWORDCASE) VALUE(ASIS)"`

JCL preferences

Use **JCL** preferences to specify JCL settings, such as JOB statement details and to define the trace information to include.

JCL preferences are identified and described in the tables that follows.

JCL

Field	**Description**
JCL Submit/Poll Timeout (secs)	The number of seconds that can pass before the Data Service Studio stops polling the host for a job to complete. If the host does not complete the job within that number of seconds, the job status is checked on the mainframe. The default setting is **300**.

Generation

Fields	**Descriptions**
Job Name Suffix	Used to generate a default job name. This one-character suffix is appended to the user ID. The default setting is **A**.
Job Account	Optional accounting information that you can add to the JOB statement. The default setting is blank.
Execution Class	The execution class to be used in the JOB statement. The default setting is **A**.
Message Class	The message class to be used in the JOB statement. The default setting is **X**.
Region Size	The region size to be used in the JOB statement. The default setting is **0M**.
Temporary DASD Name	Generic unit name to be used in the job step for use in allocating temporary work files. The default setting is **SYSDA**.

Trace/Debug

Fields	**Descriptions**
Request Status Values	Determines the type of status values to include in the trace: • **ALL** • **TERSE** • **VERBOSE** • **NONE** The default setting is **ALL**.
Level	Determines the trace level to use (1, 2, 3, or 4). The default setting is **1**.
Volume	Determines the trace volume to use (**QUIET**, **SILENT**, or **NOISY**). The default setting is **QUIET**.
Trace Function Stems	Enables the tracing of function stems. This setting disabled by default.

Fields	**Descriptions**
Dump REXX Variables	Enables the tracing of REXX dump variables. This setting is disabled by default.
Enable SSI Tracing	Enables SSI tracing. This setting is disabled by default.
Enable SSI SSOB dumps	Enables SSI SSOB dump tracing. This setting is disabled by default.

Metadata Discovery preferences

Use **Metadata Discovery** preferences to define settings for the WebSphere Application Server that hosts IBM Rational Asset Analyzer (RAA).

When using RAA to access VSAM or sequential data sets for COBOL applications, complete COBOL layout information that is required to map data is not available in the DB2 database. The mapping wizard uses a RESTful HTTP query to collect record layouts when data is mapped. While this query can be done directly to the Data Service server for data in PDS files, the preferred method to collect COBOL information is to retrieve record layouts directly from the WebSphere Application Server that hosts RAA.

Metadata Discovery preferences are identified and described in the table that follows.

Field	**Description**
RAA REST Root URL	Location of the RAA WebSphere Application Server. For example: `https://`*`<host>`*`:`*`<port>`*
Alternate User ID	User ID for the RAA WebSphere Application Server. You can leave this field blank if the credentials are the same as those used to connect to the current Data Service server (using **Set Server**).
Alternate Password	Password for the RAA WebSphere Application Server user ID. Specify a value in this field only if a user ID has been specified in the **Alternate User ID** field.

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	Maximum number of rows to return in the **SQL Results** view. The default value is `1000`.
SQL Results Max Bytes	Maximum number of data bytes to return in the **SQL Results** view. The default value is `1000000`.

Field	Description
SQL Results values accessed as	Specifies how data values are returned. Options include `String` or `Object`. The default setting is `String`.
DB2 CARDINALITY in generated UDTF query	Specifies the cardinality value for the CARDINALITY clause that is included in generated UDTF queries. Using the CARDINALITY clause can improve the performance of queries with UDTF references. The default value is `10000`. Specifying `0` omits the CARDINALITY clause from the generated query.
Use prepared statement to retrieve SQL column info for DB2 or DRDA tables	The Data Service 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 Data Service 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.
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 Data Service 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 Data Service 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.

SSL preferences

Use **SSL** preferences to secure JDBC and HTTP network communications between the Data Service Studio and the server.

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

Field	Description
Use SSL for Studio-Server communications (JDBC and HTTP)	Enables secure JDBC and HTTP network communications between the Data Service Studio and the server. If enabled, select the **Protocol** version to use for communications between the Data Service Studio and the server. The default setting is **TLS 1.2**.
Server Authentication	Select the authentication strategy to use: • **Require Server Validation**: Enable to require that all server certificates be authenticated and complete the following fields: – **Truststore**: The path name of the file on the local machine. The file must contain the server certificate authority (CA). – **Password**: The password for the truststore file. – **Type**: The truststore file type. For example, JKS, PKCS12, BKS, UBER. • **Allow Self-Signed Certificates**: Enable to allow the server to use self-signed certificates and complete the following fields: – **Truststore**: The path name of the file on the local machine. The file must contain the self-signed server CA (certificate authority) certificate. – **Password**: The password for the truststore file. – **Type**: The truststore file type. For example: JKS, PKCS12, BKS, UBER. • **Trust All**: Enable to allow all server certificates. If enabled, the Data Service Studio does not validate the server certificate. The default setting is **Require Server Validation**.
Client Authentication	To enable client authentication by the server, select **Enable Client Authentication** and complete the following fields: • **Keystore**: The path name of the file on the local machine. The file must contain a client certificate which has been signed by the server CA. • **Password**: The password for the keystore. • **Type**: The keystore file type. For example: JKS, PKCS12, BKS, UBER. • **Alias**: To confirm that the password is valid and that the alias (label) appears, click **Refresh**. This setting is disabled by default.

Accessibility

The publications for IBM Open Data Analytics for z/OS are available in IBM Knowledge Center and Adobe Portable Document Format (PDF) and comply with accessibility standards. If you experience difficulties when you use any of the information, notify IBM through one of the comment methods described in "How to send your comments to IBM" on page vii.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Trademarks

Index

SC27-9034-00