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

User's Guide

SC27-9301-00

Contents

Tables

About this information

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

Purpose of this information

This document provides an overview of IBM Data Virtualization Manager for z/OS and presents the information you need to access your data sources using the Data Virtualization Manager studio.

Who should read this information

This information is intended for system and database administrators.

How to send your comments to IBM

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

Important: If your comment regards a technical problem, see instead “If you have a technical problem” on page ix.

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

Include the following information:

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

  IBM Data Virtualization Manager for z/OS User's Guide
  SC27-9301-00
- The topic and page number or URL of the specific information to which your comment relates
- The text of your comment.

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

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

If you have a technical problem

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

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

Chapter 1. Introduction

This document provides an overview of IBM Data Virtualization Manager for z/OS and presents the information you need to access your data sources using the Data Virtualization Manager studio.

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

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

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

Description	Related APARs
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 74 and “SQL preferences” on page 99.	PH03533
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 84.	PH01593 PI99536
Db2 Direct is a new Data Virtualization Manager 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 53.	PI95751
A new option is provided to map Adabas binary fields to numeric packed decimal format. See “Creating virtual tables for Adabas data” on page 51.	PI94474
SQL query access to DB2 unload data sets is now provided. See “Accessing Db2 unload data” on page 85.	PI94369
When connecting from the Data Virtualization Manager studio to the Data Virtualization Manager server, password phrase authentication is supported. See “Connecting to the Data Virtualization Manager server” on page 47.	PI92952
SQL access to IBM MQ is now provided. See “Creating virtual tables for IBM MQ” on page 60.	PI92252
IMS Direct now supports access to multiple IMS subsystems. See “Creating virtual tables for IMS data” on page 55, which includes updated procedures.	PI90971
You can now specify a generation data group base name when defining a virtual table. See “Creating virtual tables for sequential data” on page 64.	PI90302
Improvements have been made to the z/OS Connect documentation. See “Configuring the z/OS Connect solution” on page 27 and “Connecting to z/OS Connect” on page 86.	
The list of data source types supported by Data Virtualization Manager has been updated. Information about SQL access by data source type is also provided. See “Supported data sources” on page 3.	

Components

This topic provides a brief introduction to the product components in Data Virtualization Manager.

IBM Data Virtualization Manager for z/OS consists of the following components:

Data Virtualization Manager server
The Data Virtualization Manager server is the backend engine that resides on the mainframe and processes all requests to and from the client. The server runs as started task.

Data Virtualization Manager studio
The Data Virtualization Manager studio (studio) is an Eclipse-based user interface that communicates with the Data Virtualization Manager server. Use the studio to perform the tasks required to get access to your mainframe data from your programs and services.

Web Services
Data Virtualization Manager Web Services provides access to data through RESTful APIs. Use of Data Virtualization Manager Web Services requires IBM z/OS Connect Enterprise Edition.

Terminology

This topic provides a brief introduction to the product terminology in Data Virtualization Manager.

The following terms and abbreviations appear throughout this documentation:

Configuration member
The *configuration member* (also referred to as the *server initialization member*) is a PDS member that is used by the Data Virtualization Manager server for initialization options.

Data virtualization
A process that allows access to data without regard to its location or format.

NoSQL solution
NoSQL connectivity provides access to any data source supported by Data Virtualization Manager through a robust MongoDB query language.

Server initialization member
See *configuration member*.

SQL solution
SQL connectivity provides ANSI-92 SQL access through mainframe data virtualization. You can use it to transform mainframe artifacts into a familiar, easy-to-use relational format. With industry standard SQL access, Data Virtualization Manager makes critical mainframe data readily available to business intelligence or analytic applications.

Virtual collection
A *virtual collection* is a group of MongoDB documents and is used for NoSQL access to data. A collection is similar to a relational database table, but differs in that is does not enforce a schema. A virtual collection is created using the studio and is done automatically when you create a virtual table using NoSQL access. JavaScript is generated from the virtual collection to read and extract data from the mainframe. For more information, see “Virtual collections” on page 3.

Virtual map
A *virtual map* is simply the metadata of your data source and is created using the studio. The terms *virtual table* and *virtual map* are sometimes used interchangeably.

Virtual table
A *virtual table* is a relational representation of the data source that you want to virtualize. The virtual table is used to generate and execute the SQL that reads and extracts the mapped data from the source. Virtual tables are created using the studio. For more information, see “Virtual tables” on page 3.

Virtual view
A *virtual view* can be created from virtual tables as an alternate way of viewing and accessing your data. Virtual views are created using the studio.

Virtual tables

A virtual table enables data from multiple, disconnected sources to be virtually integrated into a single logical data source. You can use both mainframe and distributed data sources to build mappings and create virtual tables regardless of where the data is located and without having to copy or move the data.

Each data source provides a mechanism for defining data. For example, for VSAM and sequential file data, you use a COBOL copybook to store data definitions. When you create a virtual table, the data definitions determine how to create the mappings.

You create a virtual table only once. After you create a virtual table, you can use any interface to access the data. Virtual tables are used for SQL solutions.

Virtual collections

Virtual collections are used for NoSQL solutions. A collection is a group of MongoDB documents. A collection is similar to a relational database table, but differs in that is does not enforce a schema.

Like virtual tables, virtual collections provide virtualization of data. To create a virtual table or virtual collection for data sources, use the studio. For more information, see Chapter 7, “The Data Virtualization Manager studio,” on page 43.

Supported data sources

The Data Virtualization Manager server supports a broad range of data sources, including mainframe relational/non-relational databases and file structures, mainframe applications and screens, distributed databases running on Linux, UNIX, and Windows platforms, cloud-based relational and non-relational data, and NoSQL databases. IBM Data Virtualization Manager for z/OS solutions have a wide range of connectivity options for data consumers, including ANSI-92 SQL (JDBC/ODBC), NoSQL (JSON), Services (SOAP/REST) and HTML.

Data Virtualization Manager supports the data source types listed in the following table.

Table 1. Supported data source types

Data support	Data source type
Mainframe relational/non-relational databases	• IBM® Db2 for z/OS • IBM® Information Management System (IMS) • IBM® MQ • CA IDMS • Software AG Adabas
Mainframe file structures	• Sequential files • Native VSAM files • z/OS File System (zFS)
Mainframe applications and screens	• IBM® CICS® • VSAM via IBM® CICS® Transaction Server • IBM® InfoSphere™ Federation Server • IBM® Query Management Facility (QMF) • IBM® System Management Facility (SMF) • Software AG Natural • z/OS system logging (SYSLOG)

Table 1. Supported data source types (continued)	
Data support	**Data source type**
Distributed data stores running on Linux, Unix, and Windows platforms	• IBM® BigInsights Hadoop • IBM® dashDB • IBM® Db2 • IBM® Db2 Big SQL • IBM® Informix • Apache Derby • Microsoft SQL Server • Oracle

SQL connectivity

SQL connectivity provides ANSI-92 SQL access through mainframe data virtualization. You can use it to transform mainframe artifacts into a familiar, easy-to-use relational format. With industry standard ANSI-92 SQL access, IBM Data Virtualization Manager for z/OS SQL makes critical mainframe data readily available to business intelligence or analytic applications.

The following table provides an overview of the supported SQL access for each data source type.

Table 2. SQL access by data source type

Data source type	SELECT	INSERT	UPDATE	DELETE	CALL statement
Adabas	Yes	Yes	Yes	Yes	n/a
Apache Derby	Yes	Yes	Yes	Yes	n/a
Big SQL	Yes	Yes	Yes	Yes	n/a
BigInsights Hadoop	Yes	Yes	Yes	Yes	n/a
CA IDMS	Yes	No	No	No	n/a
CICS COMMAREA	n/a	n/a	n/a	n/a	Yes
dashDB	Yes	Yes	Yes	Yes	n/a
Db2 Direct	Yes	No	No	No	n/a
Db2 for z/OS	Yes	Yes	Yes	Yes	n/a
Db2 for z/OS stored procedure	n/a	n/a	n/a	n/a	Yes
Db2 for Linux, Unix and Windows	Yes	Yes	Yes	Yes	n/a
Db2 for Linux, Unix and Windows stored procedure	n/a	n/a	n/a	n/a	Yes
IMS DBCTL	Yes	Yes	Yes	Yes	n/a
IMS Direct	Yes	No	No	No	n/a
IMS OTMA	n/a	n/a	n/a	n/a	Yes
Informix	Yes	Yes	Yes	Yes	n/a
InfoSphere Federation Server	Yes	Yes	Yes	Yes	n/a
Microsoft SQL Server	Yes	Yes	Yes	Yes	n/a

Table 2. SQL access by data source type (continued)

Data source type	SELECT	INSERT	UPDATE	DELETE	CALL statement
MQ	Yes	No	No	No	n/a
Natural	n/a	n/a	n/a	n/a	Yes
Oracle	Yes	Yes	Yes	Yes	n/a
QMF	Yes	No	No	No	n/a
Sequential data set	Yes	Yes	No	No	n/a
SMF	Yes	No	No	No	n/a
VSAM data set **Note:** For more information about access to different VSAM file types, see "IBM Data Virtualization Manager for z/OS Interface for VSAM and Sequential files" in the *Administration Guide*.	Yes	Yes	Yes	Yes	n/a
z/OS SYSLOG	Yes	No	No	No	n/a
zFS	Yes	No	No	No	n/a

NoSQL connectivity

NoSQL connectivity provides access to any data source supported by IBM Data Virtualization Manager for z/OS through a very robust MongoDB query language.

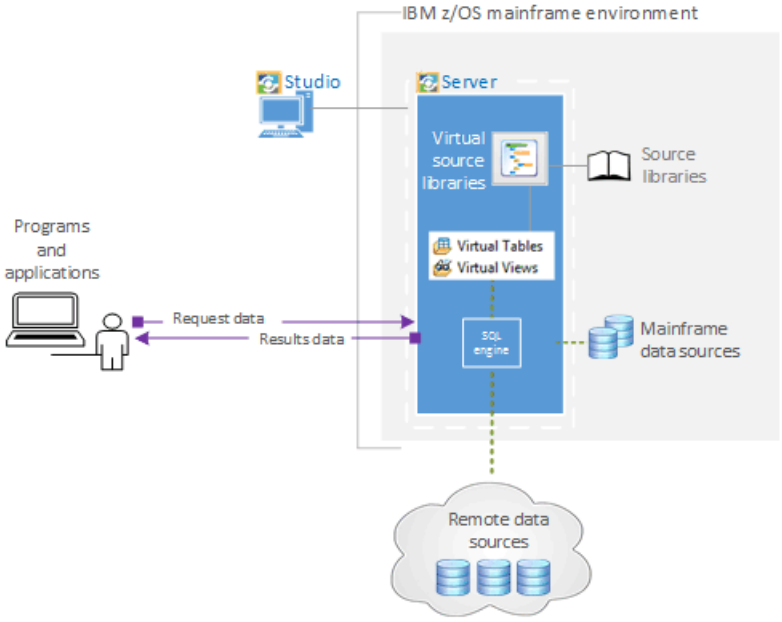
Chapter 2. Getting started

About this task

Use IBM Data Virtualization Manager for z/OS (Data Virtualization Manager) to securely connect to and access data from your z/OS mainframe environment.

This section provides an overview of the general tasks that are associated with implementing Data Virtualization Manager within your existing z/OS mainframe environment to get virtual access to your mainframe data from your applications.

The following illustration shows the key Data Virtualization Manager components within the Data Virtualization Manager architecture.



Task 1: Install and configure the Data Virtualization Manager server

About this task

Install the Data Virtualization Manager server using IBM SMP/E for z/OS.

Procedure

Use the information in *Program Directory for IBM Data Virtualization Manager for z/OS* to install the Data Virtualization Manager server on your system.

What to do next
Install the drivers.

Task 2: Install the drivers

About this task
Install the drivers on your development workstation that are required to access data made available through Data Virtualization Manager. Data Virtualization Manager supports the following drivers:

- JDBC

- ODBC

Procedure

See the *IBM Data Virtualization Manager Installation and Customization Guide* for details about installing the drivers.

What to do next
Install and configure the Data Virtualization Manager studio.

Task 3: Install and configure the Data Virtualization Manager Studio

About this task

Install and configure the Data Virtualization Manager Studio on your Windows system.

Procedure

See the *IBM Data Virtualization Manager Installation and Customization Guide* for details about installing and configuring the IBM Data Virtualization Manager studio.

What to do next
Use the Data Virtualization Manager studio to virtualize the mainframe data.

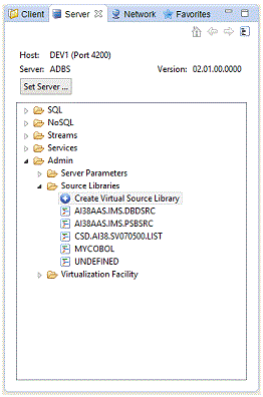
Task 4: Virtualize your mainframe data

About this task
To use SQL or NoSQL to access your mainframe data, use the Data Virtualization Manager studio to create the metadata on the Data Virtualization Manager server. The metadata is used to reference source libraries that already exist on your mainframe. The information that is required to virtualize the source data depends on the data type. For example, for a VSAM file, the virtual source library must contain the copybook that describes the structure of the records in the VSAM file. For an IMS database, you need to provide the Database Definition (DBD) and Program Specification Block (PSB) files, and a copybook structure for each segment of the IMS database that you want to virtualize.

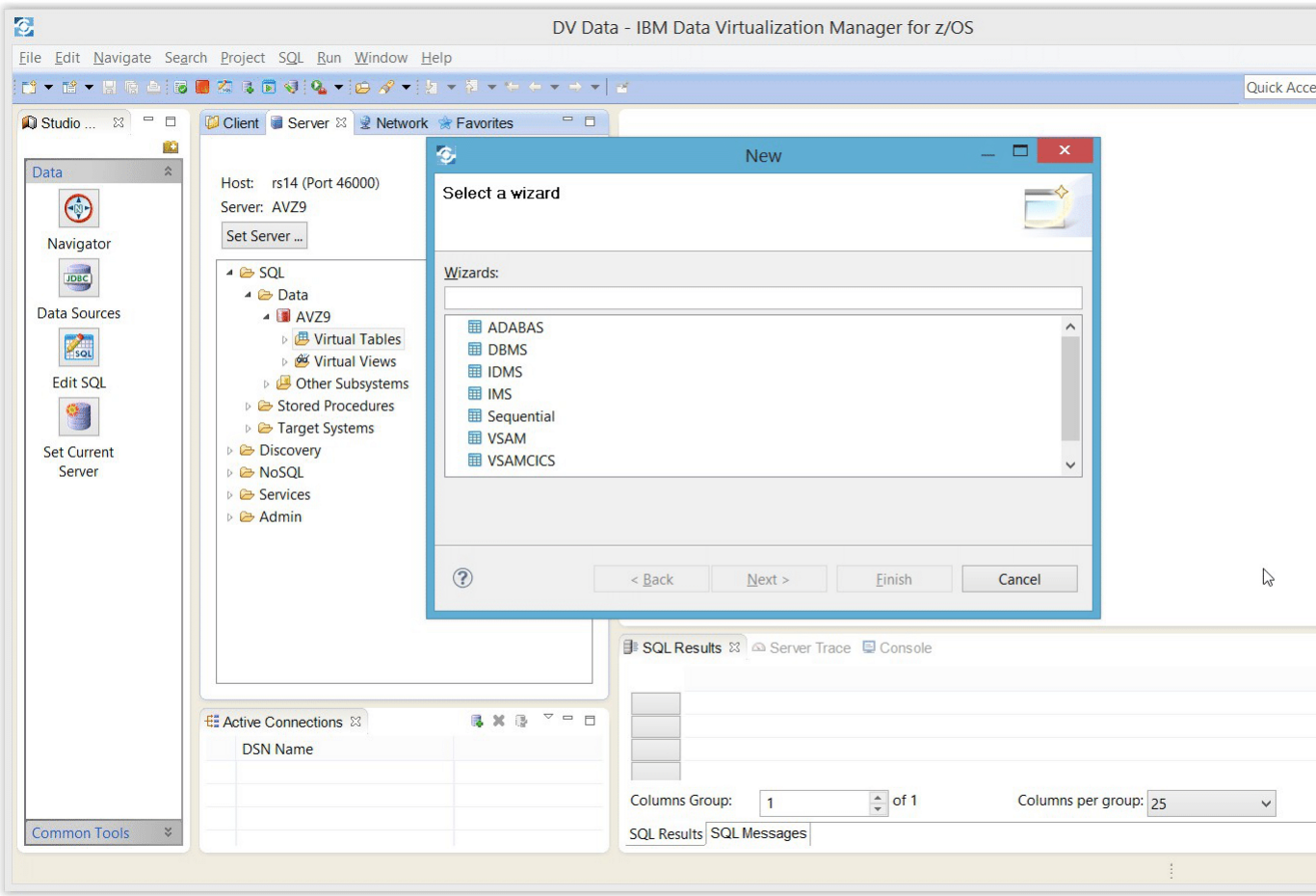
Note: For complete task details, see “Creating server metadata” on page 49.

Procedure

1. Use the Studio to connect to the Data Virtualization Manager server.
2. Use the **Create Virtual Source Library** wizard to create virtual source libraries that reference the data layouts on the mainframe.



3. Create a virtual table (virtual representation) that maps to the data that you want to access:



- For SQL access to data, complete the virtual table wizard that is appropriate for the data that you want to access.
- For NoSQL access to data, complete the wizard that is appropriate for the data that you want to access.

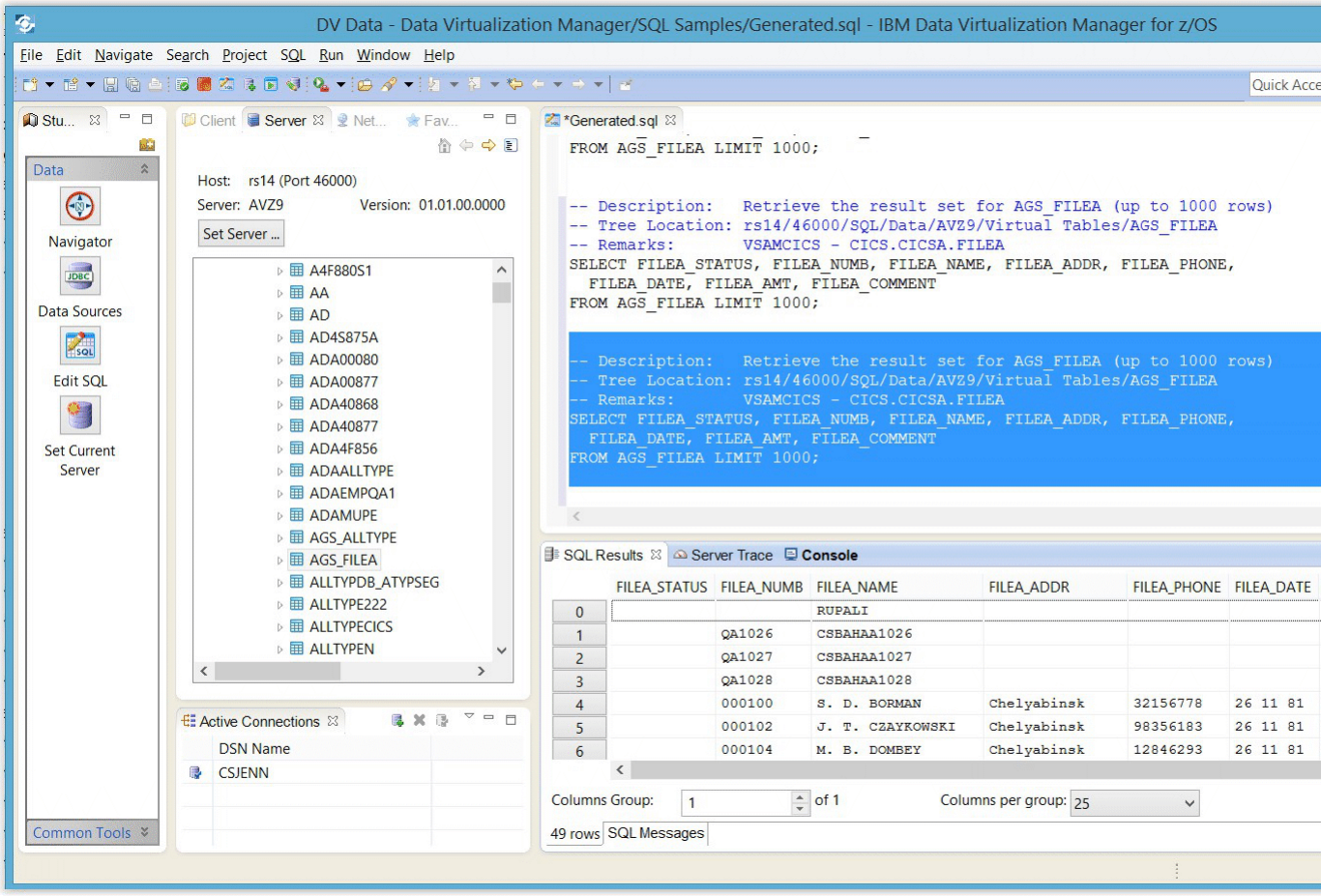
What to do next
Generate and execute the SQL that is used to test access to the data on the mainframe and to review the resulting data.

Task 5: Generate and execute a query

About this task
To test access to your data, generate and execute a query from an existing virtual table.

Procedure

1. On the Studio **Server** tab, right-click the virtual table that maps to the data that you want to access and select **Generate Query**.
2. Depending on your preference settings, the query is automatically executed after being generated or you are prompted to execute the statement. If prompted, click **Yes**.



3. In the **SQL Results** view, review the result set that displays. If necessary, in the **Generated.sql** view, modify the SQL to select only the data that you want to access. Any ANSI compliant SQL is acceptable.

What to do next
You can now generate the code from the SQL.

Task 6: Generate sample code

About this task
To access mainframe data from your programs and applications, use the Studio to generate sample code.

Note: For task details, see the "Generating code from SQL" on page 82.

Generate sample code for SQL access to your data

Procedure

1. Right-click the virtual table or the selected SQL statement in the **Generated.sql** view and select **Generate Code From SQL**.
2. Complete the **Code** wizard to generate the code for the programming language/environment that you want to use. Choices include:
   - **Java JDBC Class**
   - **Java Spark Application**
   - **Scala Jupyter Notebook**

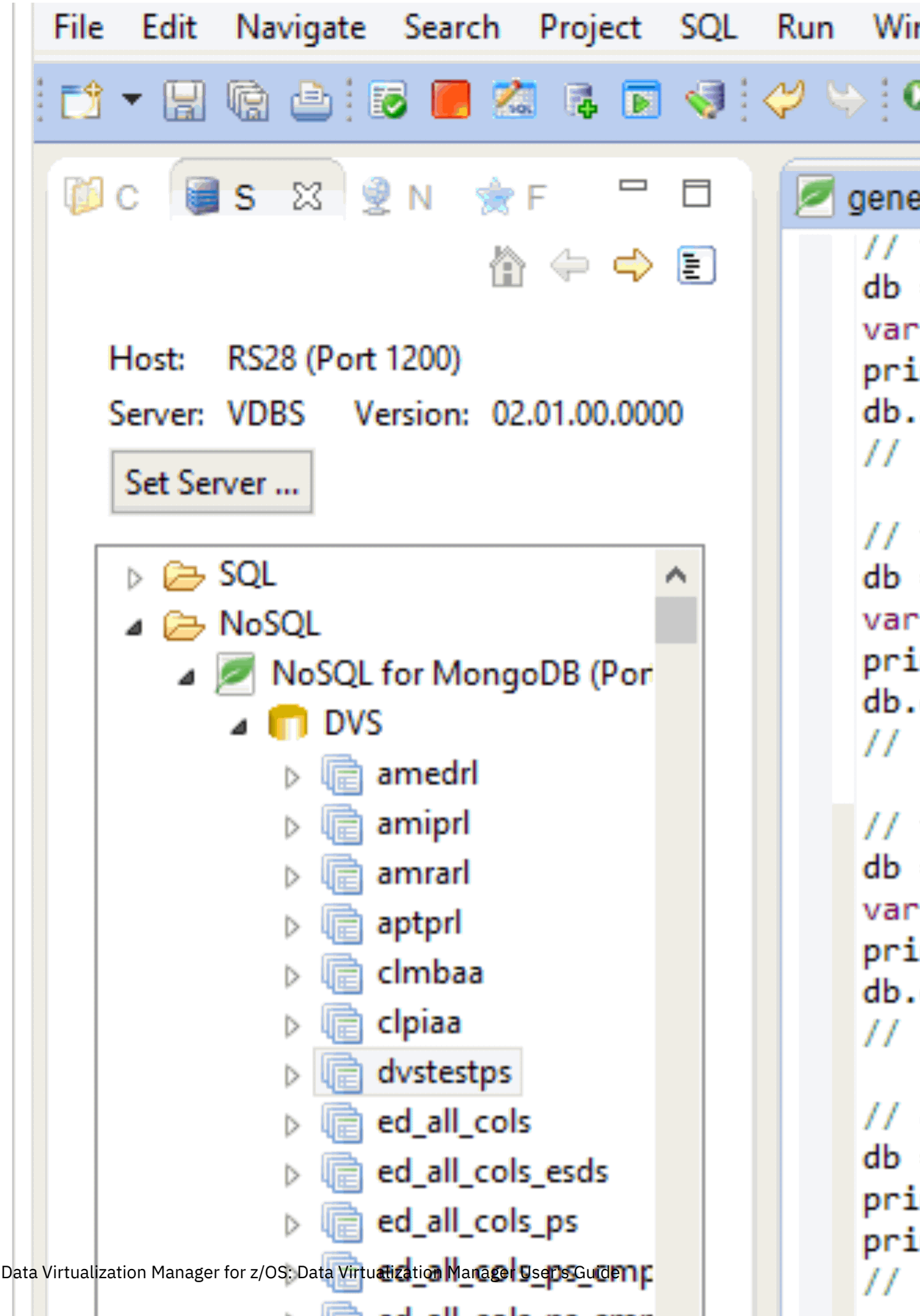
Note: The generated Scala Jupyter Notebook code provides a simple starter program showing you how to use the JDBC driver to load data into a Spark application. The wizard does not allow SQL parameter markers for this application type.

- **Scala Spark Application**
- **VB.NET Class**
- **C# Class**
- **Other** (requires that you specify XSLT file and file extension)

Generate sample code for NoSQL access to your data

Procedure

1. Select the virtual collection that represents the data you want to access.
2. Select **Generate Query** and select either **find()**, **find(...)**, or **count()**.
3. In the ***generated.js** view, right-click the selected query and select **Execute Mongo Script**.


File Edit Navigate Search Project SQL Run
Host: RS28 (Port 1200)
Server: VDBS Version: 02.01.00.0000
Set Server ...
SQL
NoSQL
NoSQL for MongoDB (Por
DVS
amedrl
amiprl
amrarl
aptprl
clmbaa
clpiaa
dvstestps
ed_all_cols
ed_all_cols_esds
ed_all_cols_ps

Results
Use the generated sample code in your programs and applications.

Chapter 3. The SQL solution

To enable ANSI SQL access to mainframe data sources, configure IBM Data Virtualization Manager for z/OS.

This SQL solution provides access to the following mainframe data sources:

- Software AG Adabas
- IBM® DB2
- IBM® Information Management System (IMS/DB)
- Native VSAM files
- Sequential files
- VSAM via IBM® CICS® Transaction Server

You must configure one or more data sources to use the Data Virtualization Manager solution.

Configuring a solution can include one or more of the following tasks:

- Configure the started task JCL by modifying the AVZS (subsystem default ID) member that is in the *hlq*.SAVZCNTL library.
- Configure the server member that is included in *hlq*.AVZS.SAVZEXEC(AVZSIN00). The Data Virtualization Manager configuration member is shipped in data set member *hlq*.SAVZEXEC(AVZSIN00) and copied to *hlq*.AVZS.SAVZEXEC(AVZSIN00) by the job in the AVZGNMP1 member for you to make your local modifications. See "Creating system data sets" in the *Installation and Customization Guide*.
- Make definition changes in the data provider interface.

Before configuring the SQL solution, the Data Virtualization Manager server installation must be successfully completed.

You can also use the Data Virtualization Manager studio to get SQL access to your data. For details about accessing data using SQL, see Chapter 7, “The Data Virtualization Manager studio,” on page 43.

For information about configuring the SQL solution, see "Configuring access to data sources in the *IBM Data Virtualization Manager for z/OS Installation and Customization Guide*.

Chapter 4. The NoSQL solution

The NoSQL solution enables developers to write applications that use a JSON-oriented query language that is created by MongoDB to interact with the virtualized data.

JSON (JavaScript Object Notation) is a lightweight data-interchange format that uses readable text to transmit data. JSON is an open standard and provides an alternative to XML. For more information, see https://www.ietf.org/rfc/rfc4627.txt.

To configure the NoSQL interface for Mongo, you need to modify the Data Virtualization Manager configuration member.

After you establish a connection from a MongoDB client, all supported MongoDB methods can be used to query and update the data that is defined to the server. This exposes existing data as JSON documents and communicates with Mongo applications. Since this solution is about access to existing data, not all methods in MongoDB query language are applicable.

To configure and verify access to data by using the MongoDB query language:

- Modify the Data Virtualization Manager configuration member
- Configure data sources

Modifying the Data Virtualization Manager configuration member

To start using the MongoDB query language with the Data Virtualization Manager server, enable the **MONGODB**, **MONGOPORT**, and **MONGOZSQLDBNAME** parameters and connect to the server with your choice of MongoDB client.

Before you begin

The server must be installed.

About this task

Configure a data source that is supported by the Data Virtualization Manager server, create your source libraries, and map the virtual collections.

The Data Virtualization Manager configuration member is shipped in data set member *hlq*.SAVZEXEC(AVZSIN00) and copied to *hlq*.AVZS.SAVZEXEC(AVZSIN00) by the job in the AVZGNMP1 member for you to make your local modifications.

Procedure

1. In the AVZSIN00 member, locate the comment “ENABLE MONGO API.”
2. To enable the MongoDB parameters, change `if DontDoThis` to `if DoThis`.

```
if DoThis then
  do
        "MODIFY PARM NAME(MONGODB)              VALUE(YES)"
        "MODIFY PARM NAME(MONGOPORT)            VALUE(1207)"
        "MODIFY PARM NAME(MONGOZSQLDBNAME)      VALUE(DVS)"
end
```

The following table lists the required parameters:

Parameter	Description	Valid values
MONGODB	Activates support for MongoDB query language	**NO** Support is not active. **YES** Activate support.
MONGOPORT	TCP/IP port number.	Numeric value. Default is 1207.
MONGOZSQLDBNAME	Database name for SQL engine requests.	Text string. Default is “AVZ”.

The following table lists optional parameters:

Parameter	Description	Valid values
MONGOSSLPORT	TCP/IP main SSL port number.	Default is 0.
MONGOTRACELEVEL	Text trace message verbosity.	Default is normal.
MONGOCURSORLIFE	Cursor wait time limit in seconds.	Default is 600 seconds.
MONGOMAXMSGLENGTH	Maximum message length, in MB.	Default is 48 MB.

MongoDB interface examples

JavaScript, C#, and BIRT interface examples show how to use MongoDB query language with the Data Virtualization Manager server.

NoSQL for MongoDB interface example

The following copybook describes a VSAM file that contains information about zip codes and cities.

```
************************************************
* RECORD LAYOUT
*
 01 RECORD.
    05 ZIP                        PIC X(5).
    05 CITY                       PIC X(16).
    05 STATE                      PIC X(2).
    05 POPULATION                 PIC 9(6).
    05 LOC_X                      PIC 9(3)V9(6).
    05 LOC_Y                      PIC 9(2)V9(6).
```

This is a sample of how data looks in the VSAM file:

```
****  Top of data  ****
01001AGAWAM          MA01533807262273942070206
01002CUSHMAN         MA03696307251565042377017
01005BARRE           MA00454607210835442409698
01007BELCHERTOWN     MA01057907241095342275103
01008BLANDFORD       MA00124007293611442182949
01010BRIMFIELD       MA00370607218845542116543
01011CHESTER         MA00168807298876142279421
```

After you virtualize this VSAM structure in a virtual collection, you can retrieve this information as a JSON document by using MongoDB query language. This example uses the MongoDB shell:

```
C:\mongodb-win32-x86_64-2.4.5\bin>mongo host:27017
MongoDB shell version: 2.4.5
connecting to: host:27017/test

> show dbs
DVS     1GB
```

```
> use DVS
switched to db DVS

> db.us_zipcodes.find({"city":"AGAWAM"})


{
        "zip" : "01001",
        "city" : "AGAWAM",
        "state" : "MA",
        "population" : 15338,
        "loc_x" : 72.622739,
        "loc_y" : 42.070206
}
```

Where:

- database – similar to schema in SQL, in the example “DVS”
- collection – similar to the table, in the example “us_zipcodes”
- find() method – similar to SQL SELECT statement with predicate specified in JSON format

JavaScript and Node.js

Download the MongoDB JavaScript driver from the MongoDB website or use the Mongo shell to run JavaScript.

The following is an example of how to use Node.js with the NoSQL interface for Mongo.

```
/*jshint node:true*/

// app.js
// This file contains the server side JavaScript code for your application.
// This sample application uses express as web application framework
// (http://expressjs.com/), and jade as template engine (http://jade-lang.com/).

var express = require('express');
var mongo = require('mongodb');

var mongonative = require('mongodb').MongoClient,
   assert = require('assert');

var dbURI = "mongodb://dvs_host_name:port/db_name";

// setup middleware
var app = express();
//app.use(app.router);
//app.use(express.errorHandler());
app.use(express.static(__dirname + '/public')); //setup static public directory
app.set('view engine', 'jade');
app.set('views', __dirname + '/views'); //optional since express defaults to
  CWD/views

// render index page
app.get('/', function (req, res) {
    res.render('index');
});

mongonative.connect(dbURI, function (err, db) {
    console.log("connect with native mongodb driver");
    assert.equal(null, err);
    console.log("connected with native mongodb driver");

    app.get('/docs', function (req, res) {
        console.log('In GET /docs');
        var collection = db.collection('us_zipcodes');
        var adminDb = db.admin();
        if (err) throw err;

        /*adminDb.ping(function (err, pingResult) {
            console.log('In admindb functions: ', pingResult);
            assert.equal(null, err);
            db.close();
        });*/

       /* adminDb.listDatabases(function (err, dbs) {
            console.log('In admindb listdbs: ', dbs);
            assert.equal(null, err);
```

```
            assert.ok(dbs.databases.length > 0);

            db.close();
        }); */

       /* adminDb.command({buildInfo:1}, function(err, info) {
            console.log('In buildinfo: ', info);
            assert.equal(null, err);
            db.close();
        });*/

        // Return the information of all collections, using the callback format
      /*  db.collectionsInfo(function (err, cursor) {
            // Turn the cursor into an array of results
            cursor.toArray(function (err, items) {
                assert.ok(items.length > 0);
                var json = JSON.stringify(items);
                res.end(json);
                db.close();
            });
        });*/
       /* db.collectionNames(function (err, items) {
                assert.ok(items.length > 0);
                var json = JSON.stringify(items);
                res.end(json);
                db.close();
        });*/

        //db.collections(function (err, docs) {
        //doesn't work with DVS, use above instead:
        //collection.find({}).toArray(function (err, docs) {
        //collection.count(function (err, docs) {
        //collection.find({zip:"01001"}).toArray(function (err, docs) {
        //collection.distict(city, function (err, docs) {
        //collection.findOne(function (err, docs) {
        //collection.find({ zip: { "$gte": "96700", "$lte": "96799" } },
           { "sort": "population" }).toArray(function (err, docs) {
        collection.find({ zip: { "$in": ["96701", "96702", "96703"] } }).
         toArray(function (err, docs) {

         console.log('In find city: ', docs);
         assert.equal(err, null);
         var json = JSON.stringify(docs);
         res.end(json);
         db.close();
         });
    });
});


// There are many useful environment variables available in process.env.
// VCAP_APPLICATION contains useful information about a deployed application.
var appInfo = JSON.parse(process.env.VCAP_APPLICATION || "{}");
// TODO: Get application information and use it in your app.

// VCAP_SERVICES contains all the credentials of services bound to
// this application. For details of its content, please refer to
// the document or sample of each service.
var services = JSON.parse(process.env.VCAP_SERVICES || "{}");
// TODO: Get service credentials and communicate with bluemix services.

// The IP address of the Cloud Foundry DEA (Droplet Execution Agent)
   that hosts this application:
var host = (process.env.VCAP_APP_HOST || 'localhost');
// The port on the DEA for communication with the application:
var port = (process.env.VCAP_APP_PORT || 3000);
// Start server
app.listen(port, host);
console.log('App started on port ' + port);
```

C#

Download and install the C# driver from the MongoDB website.

The following is a sample class that calls MongoDB:

```
using MongoDB.Bson.IO;
using MongoDB.Bson.Serialization;
using MongoDB.Bson.Serialization.Attributes;
using MongoDB.Bson.Serialization.Conventions;
```

```
using MongoDB.Bson.Serialization.IdGenerators;
using MongoDB.Bson.Serialization.Options;
using MongoDB.Bson.Serialization.Serializers;
using MongoDB.Driver.Wrappers;

namespace MongoClass
{
    public class MongoClassWrapper
    {
        public string mongoreturns;
        public MongoClassWrapper(string host, string database_name,
                    string collection_name, IMongoQuery query)
        {
            MongoServerSettings serversettings = new MongoServerSettings();
                    serversettings.To invoke from your
                        .NET application use following:GetDatabase(database_name);
            try
            {
                MongoCursor testcursor =
                            database.GetCollection(collection_name).Find(query);
                mongoreturns = testcursor.ToJson();
            }
            catch (Exception ex) { }
            finally
            {
                server.Disconnect();
            }

        }
    }

}
```

Eclipse BIRT

You can use the Business Intelligence Reporting Tool (BIRT) to create reports from a MongoDB data source.

Before you begin

- Eclipse Kepler (minimum) http://www.eclipse.org/kepler/
- BIRT (Business Intelligence and Reporting Tools) reports http://download.eclipse.org/birt/downloads/

Procedure

1. Install BIRT into an existing Eclipse Kepler. In Eclipse, select **Help** > **Install New Software**, select BIRT update site: http://download.eclipse.org/birt/update-site/4.3

   Note: If you download BIRT with KEPLER from the BIRT site, you can skip this installation step.
2. Restart Eclipse, and follow the *BIRT Report Developer Guide*.
3. Create a new project and then new report (select blank report).
4. Add data source – select MongoDB data source.
5. Create a new data set by right clicking on DataSet node. Drag and drop the data set onto report.
6. Select **Preview** tab on the bottom of the report to run a query.

Supported MongoDB query language

Data Virtualization Manager server supports the following MongoDB query language collection, cursor, operator, aggregation, and administrative functions.

Collection

MongoDB	Example

count()
`<collection>.count()`
	`db.us_zipcodes.count()`
	or database command:
	`db.runCommand({count:"us_zipcodes"})`

distinct()
`<collection>.distinct(<field>,<query>)`
	`db.us_zipcodes.distinct("city",{"state":"GA"})`
	or database command:
	`db.runCommand({ distinct: "us_zipcodes", key: "city", query: { "state": "GA" } })`

find() `<collection>.find(<query>,<fields>)`	`db.us_zipcodes.find( {"state": "CA"}, {"zip": 1, "city": 1}).sort({"zip": -1})`
findOne() `<collection>.findOne(<query>,<fields>)`	`db.us_zipcodes.findOne( {"state": "CA"}, {"zip": 1, "city": 1})`
getIndexes() `<collection>.getIndexes()`	`db.us_zipcodes.getIndexes()`
insert() `<collection>.insert`	`db.us_zipcodes_update.insert({ "zip": "02001", "city" : "AGAWAM", "state" : "MA", "population" : 15338, "loc_x" : 72.622738, "loc_y" : 42.070205});`
remove() `<collection>.remove`	`db.us_zipcodes_update.remove({ "zip": "02001" });`
update() `<collection>.update`	`db.us_zipcodes_update.update({ "zip": "02001" }, { $set: { "city" : "CUSHMAN", "population" : 36709, "loc_x" : 72.51565, "loc_y" : 42.377016999999995 } }, false, false);`
drop() `<collection>.drop`	Not supported
ensureIndexes() `<collection>.ensureIndexes`	Not supported The `collection.EnsureIndex()` API allows to create document indexes on fields. In this version, rely only the primary keys/indexes defined in data sources.

MongoDB	Example
save() <collection>.save	Not supported In MongoDB the save() document API at the collection level does not rely on preexistence of the collection. At the first insert() or save(), if the collection does not exist MongoDB server allocate a new one based on the collection name. Since NoSQL relies on existing data set that we do not support the dynamic creation of collection

Cursor

MongoDB	Example
count() <cursor>.max(<IndexBounds>)	db.us_zipcodes.find({"state": "CA", "city": "LOS ANGELES"}).count()
limit() <cursor>.limit(<n>)	db.us_zipcodes.find().limit(10);
max() <cursor>.max(<IndexBounds>)	db.us_zipcodes.find({"state": "CA"}, {"zip": 1, "city": 1}).sort({"zip": -1})
findOne() <collection>.findOne(<query>,<fields>)	cursor = db.us_zipcodes.find({ "city": "IRVINE" }).max({ "zip": "92718" }) {"zip" : "16329", "city" : "IRVINE ", "state" : "PA", "population" : 479, "loc_x" : 79.28630299999999, "loc_y" : 41.843283 } {"zip" : "92714", "city" : "IRVINE ", "state" : "CA", "population" : 60654, "loc_x" : 117.79892799999999, "loc_y" : 33.6876408 } {"zip" : "92715", "city" : "IRVINE ", "state" : "CA", "population" : 30690, "loc_x" : 117.82125099999999, "loc_y" : 33.6508841 } SELECT * FROM shadow.us_zipcodes WHERE (city = 'IRVINE' AND zip<'92718')
min() <cursor>.min(<IndexBounds>)	cursor = db.us_zipcodes.find({ "city": "IRVINE" }).min({ "zip": "92718" }) {"zip" : "92718", "city" : "IRVINE ", "state" : "CA", "population" : 1, "loc_x" : 117.71147599999999, "loc_y" : 33.6581796 } {"zip" : "92720", "city" : "IRVINE ", "state" : "CA", "population" : 23474, "loc_x" : 117.76553299999999, "loc_y" : 33.7074953 }
skip() <cursor>.skip(<n>)	db.us_zipcodes.find({ "city": "IRVINE" }).skip(1)
sort() <cursor>.sort()	db.us_zipcodes.find({"state": "CA"}, {"zip": 1, "city": 1}).sort({"zip": -1})

Operators

MongoDB	Example
Comparison	
greater than: $gt less than: $lt greater than or equal to: $gte less than or equal to: $lte <cursor>.find({field:{$gt:value}})	db.us_zipcodes.find({"zip":{"$gte":"96700", "$lte":"96799"}}).sort({"population":1}) db.us_zipcodes.find({ "zip": { "$gt": "96700", "$lt": "96799" } }).sort({ "population": 1 })
$in <cursor>.find({field:{$in: [value1,value2,value3...]}})	db.us_zipcodes.find({ "zip": { $in: [96701,96702,96703]}})
$ne Selects the documents where the value of the field is not equal to the specified value. <cursor>.find({field:{$ne:value}})	db.us_zipcodes.find({ "state": { $ne: "AL" } }, {"state":1}).limit(10)
$nin Selects the documents where the field value is not in the specified array, or the field does not exist. <cursor>.find({field:{$nin: [value1,value2,value3...]}})	db.us_zipcodes.find({ "zip": { $nin: [96701, 96702, 96703] } }).limit(10)
Logical	
$and {$and:[{expression1},{expression2},...]}	db.us_zipcodes.find({ $and: [{"population": { $gt: 30000}}, {"state":"MS"}]}).sort({ "population": 1})
$or {$or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }	db.us_zipcodes.find({ "state": "MS", $or: [{ "population": { $gt: 35000 } }, { "loc_y": { $gt: 34 }}] })
$not {field: { $not: { <operator-expression> } } }	db.us_zipcodes.find({ $and: [{"population": { $not:({ $gt: 30000 }) }, { "state": "MS" }] }).sort ({"population": 1})
Element $exist $type	Not supported
Evaluation $mod $regex $where	Not supported
Geospatial	Not supported

Aggregation

Limitations: Only particular combinations of pipeline operators are allowed. For example, "$project" or "$limit" cannot follow "$group".

Example:

```
db.us_zipcodes.aggregate([
    {$match: {loc_x: {$gte: 90.89}}},
    {$project: {
      state: "$state",
      population_thnd: {$divide: ["$population", 1000]}}
    },
    {$group: {
      _id: "$state",
      population_thnd: {$sum: "$population_thnd"}}
    },
    {$sort: {_id: 1}},
    {$skip: 1},
    {$limit: 2}
  ])
```

Administrative functions

Supported?	MongoDB	Example
No	compact	
No	text	
No	mapReduce	
Yes (partially)	getLastError	`db.getLastError()` or `db.runCommand({ getLastError: 1 })`
No	Sharding Commands	
Yes	Authentication Commands	
No	Geospatial Commands	
Yes	whatsmyuri	`db.runCommand({whatsmyuri: 1})`
Yes	resetError	`db.runCommand({resetError: 1})`
Yes	isMaster()	`db.isMaster()`
Yes	listDatabases()	`db.adminCommand("listDatabases")` or `db.runCommand( {listDatabases: 1})`
Yes (limited)	db.system.namespaces.find()	`db.system.namespaces.find()`
Yes	ping	`db.runCommand( { ping: 1 })`
Yes	buildinfo	`db.runCommand( { buildinfo: 1 })`
Partially (see aggregation)	group	`db.runCommand({ group: { ns: <namespace>, key: <key>, $reduce: <reduce function>, $keyf: <key function>, cond: <query>, finalize: <finalize function> } })`
No	drop	
No	getCmdLineOpts	

Supported?	MongoDB	Example
No	fsync	
No	filemd5	
No	collMod	
Yes	resync	
Yes	getPrevError	
Yes	replSet*	`db.runCommand({ replSetGetStatus: 1, forShell: false/true })`
Yes	isdbgrid()	`db.runCommand( { isdbgrid: 1 })`
No	renameCollection	
No	dropIndexes	
Yes (partially)	setParameter and getParameter	
Yes	connPoolStats	
Yes (partially)	cursorInfo	
	dataSize	
Yes (partially)	aggregate	
	profile	
Yes	listCommands	
Yes (partially)	getLog	
No	top	
Yes (partially)	hostInfo	
Yes (partially)	serverStatus	
Yes (partially)	dbStats()	`db.runCommand({ dbStats: 1, scale: 1})`
Yes (partially)	collStats	`db.runCommand({ collStats: 1, scale: 1})`
No	shutdown	
No	findAndModify	

Chapter 5. The z/OS Connect solution

IBM Data Virtualization Manager for z/OS provides IBM z/OS Connect Enterprise Edition direct access to data with Select, Insert, Update and Delete functions using its RESTful interface. Data Virtualization Manager uses its web service data integration created via the Data Virtualization Manager studio that is invoked via z/OS Connect and its RESTful interface.

Any data that has been virtualized via Data Virtualization Manager is available to z/OS Connect. Data Virtualization Manager deploys its Service Provider into the Unix System Services (USS) environment on Data Virtualization Manager to establish communications via WebSphere Optimized Local Adapter (WOLA) to a Data Virtualization Manager server instance in USS.

Configuring the z/OS Connect solution

To configure z/OS Connect, you need to configure the Data Virtualization Manager Service Provider on zcEE, the server started task JCL, and the configuration member.

Before you begin

The following products must be installed:

- IBM z/OS Connect Enterprise Edition

  A WebSphere Liberty angel STC must be running for communications between the zcEE server and the Data Virtualization Manager server via WOLA. For zcEE installation information, see https://www.ibm.com/support/knowledgecenter/en/SS4SVW_2.0.0/com.ibm.zosconnect.doc/installing/installing.html

  Note: Make note of the directory path for your IBM z/OS Connect Enterprise Edition installation. You will use this directory path to modify the instructions given in the examples.
- Data Virtualization Manager server
- Data Virtualization Manager studio
- The data source(s) must be configured and virtualized in Data Virtualization Manager server (see "Configuring access to data sources" in the *Installation and Customization Guide*.)

About this task

For information about configuring the z/OS Connect solution, see the following topics.

1. “Configuring the Data Virtualization Manager Service Provider on zcEE” on page 27
2. “Creating an IBM z/OS Connect Enterprise Edition server instance” on page 28
3. “Configuring the server started task JCL” on page 29
4. “Modifying the Data Virtualization Manager configuration member” on page 30
5. “Starting the Data Virtualization Manager server and z/OS Connect server” on page 31

Configuring the Data Virtualization Manager Service Provider on zcEE

You must configure the Data Virtualization Manager Service Provider on zcEE in z/OS Connect to provide communication between z/OS Connect and the Data Virtualization Manager server.

About this task

This procedure uses the Data Virtualization Manager Service Provider on zcEE installation file, which is shipped in *hlq*.SAVZBIN(AVZBIN4) and is renamed `com.rs.dv.zosconnect.provider.feature_1.0.0.esa`.

Procedure

1. From the mainframe, transfer the member *hlq*.SAVZBIN(AVZBIN4) to your development workstation using the File Transfer Protocol (FTP) in binary mode.
2. Rename the file to `com.rs.dv.zosconnect.provider.feature_1.0.0.esa`.
3. Copy the file to a UNIX System Services (USS) directory using an FTP utility.
4. Connect to UNIX System Services (USS) and install the Data Virtualization Manager Service Provider on zcEE, as follows:

   a) The **installUtility** is found in the z/OS Connect installation directory /*<zosconnect_install>*/wlp/bin, where *<zosconnect_install>* is the path directory for your z/OS Connect installation.

   For example, change to the following directory:

   ```
   cd /usr/lpp/IBM/zosconnect/v2r0/wlp/bin
   ```

   b) Set JAVA_HOME= environment variable to the path to your 64-bit IBM Java SDK.

   For example:

   ```
   export JAVA_HOME=/my_directory_path/java/IBM/J8.0_64
   ```

   c) Set WLP_USER_DIR= environment variable to the location where you want your server instances and user features to be stored.

   For example:

   ```
   export WLP_USER_DIR=/my_directory_path/var/zosconnect
   ```

   d) Issue the install command as follows:

   ```
   ./installUtility install
   /my_directory_path/com.rs.dv.zosconnect.provider.feature_1.0.0.esa
   ```

   The output from the install command looks like this:

   ```
   Step 1 of 2: Installing dvsProvider ...
   Step 2 of 2: Cleaning up temporary files ...

   All assets were successfully installed.

   Start product validation...
   Product validation completed successfully.
   ```

   Note: To uninstall the Service Provider on zcEE (for example, before installing a new version), use the **installUtility** like this:

   ```
   ./installUtility uninstall com.rs.dv.zosconnect.provider.feature
   ```

The Service Provider on zcEE has been successfully installed into z/OS Connect.

Creating an IBM z/OS Connect Enterprise Edition server instance

To prepare to use the z/OS Connect solution, create an IBM z/OS Connect Enterprise Edition server instance.

About this task

Perform this task to create a z/OS Connect server instance in UNIX System Services (USS).

As part of this task, the z/OS Connect server instance must have read/write authority to the directories and files where the server was created. There are various ways to handle USS file authority. One method, which is provided in the following procedure, is to change the authority of the z/OS Connect server instance to all directories and files so that the GROUP has read/write authority. (The z/OS Connect server

instance will run under the same GROUP defined to these directories/files.) It is recommended that you define your USS file authority according to your company requirements or standards.

Procedure

1. Perform security setup as needed for your server environment.
2. Connect to UNIX System Services (USS), and change the directory to the `/bin` directory. For example: `cd /usr/lpp/IBM/zosconnect/v2r0/bin`.
3. Create the z/OS Connect server instance by issuing the following command:

   ```
   zosconnect create <server_name> --template=zosconnect:default
   ```

   This creates the z/OS Connect server instance in `/var/zosconnect/servers`. This is a folder in the directory with your <*server_name*>.

   Note: See the IBM documentation for more details on how to create a z/OS Connect server.
4. Change the authority of the z/OS Connect server instance to have read/write access to all directories and files where the server was created, which you can do by issuing the following command:

   ```
   chmod -R g+w <top-level-server-dir>
   ```

   For example:

   ```
   chmod -R g+w /var/zosconnect/servers/avz1test
   ```

   where `avz1test` is the *server_name*.

   Note: It is recommended that you define your USS file authority according to your company requirements or standards.

   The z/OS Connect server instance is ready for configuration.
5. Modify the `server.xml` file, as follows:
   a) Locate the `server.xml` file in the following directory: `/var/zosconnect/servers/<server_name>/server.xml`
   b) Define the ports that the z/OS Connect server will listen on. These ports need to be unique.
      For example:

      ```
      httpPort="12080" httpsPort="12081"
      ```

   c) Define `wolaGroup="your_wola_name"`. The value must match your `WNAME` value in your AVZSIN00 file. This is the name the z/OS Connect server will run under in SDSF.
   d) Define `registerName="your_registry_name"`. The value must match your RNAME value in your AVZSIN00 file. There are two locations where the `registerName=` needs to be defined in the `server.xml` file.

   The z/OS Connect server instance is now ready.

Configuring the server started task JCL

Create a WebSphere Optimized Local Adaptor (WOLA) PDSE data set and copy UNIX System Services (USS) modules into it. The WOLA data set contains the modules used to communicate between a Data Virtualization Manager server and a z/OS Connect server.

Procedure

1. Allocate a PDSE data set with 30 tracks. Using the Data Virtualization Manager server naming conventions is recommended. For example: *HLQ*`.WOLA.API.V2.NA`

   ```
   Average record unit
   Primary quantity . . 30
   Secondary quantity . 2
   Directory blocks . . 15
   ```

```
Record format . . .  U
Record length . . .  0
Block size . . . . . 32760
Data set name type . LIBRARY
```

2. Change directory to /<zosconnect_install>/wlp/clients/zos. For example:

```
cd /usr/lpp/IBM/zosconnect/v2r0/wlp/clients/zos
```

3. Issue this command to copy modules from USS into PDSE:

```
cp -Xv ./* "//'<data.set>'"
```

Where <data.set> would be the data set you allocated. For example: *HLQ*.WOLA.API.V2.NA

Note: You could have several z/OS Connect servers and/or have several Data Virtualization Manager servers and they could all use the one WOLA PDSE data set.

4. Add the WOLA PDSE data set you created to the AVZRPCLB ddname in the server started task JCL.

For example:

```
//AVZRPCLB DD   DISP=SHR,DSN=&HLQ..SAVZRPC
//         DD   DISP=SHR,DSN=&HLQ.WOLA.API.V2.NA
```

Modifying the Data Virtualization Manager configuration member

Configure the z/OS Connect parameters in the AVZSIN00 configuration member. This enables communication between the Data Virtualization Manager server and the z/OS Connect server.

Procedure

Configure the z/OS Connect parameters.

You need to add a **DEFINE ZCPATH** command which defines the path to the z/OS Connect server. For example:

```
if DoThis then
   do
      "DEFINE ZCPATH",
      " NAME(your_zconnect_server_name)",
      " RNAME(your_registry_name)",
      " WNAME(your_wola_name)"
```

a) Make sure the RNAME matches the registerName in your server.xml file. This is in two locations in the server.xml file and will need to be changed in both.

b) Make sure the WNAME matches the wolaGroup in your server.xml file.

Parameter	Description	Valid values
NAME(*defname*)	This required parameter is used to provide a specific definition name.	A definition name may be up to 20 characters long, the first character must be alpha or national, and the rest must be alphanumeric or national, and the '_' and '.' are also allowed. The value is always treated as uppercase.

Parameter	Description	Valid values
RNAME(*register-name*)	This required parameter is used to supply the register name for the connection to z/OS Connect. The name supplied here must match the RegisterName value on the localAdaptersConnectService description in the z/OS Connect `server.xml` file.	A register name may be up to 12 characters long, and the first character must be alpha or national, and the rest must be alphanumeric or national. The '_' and '.' are also allowed. The value is *not* treated as uppercase. **Note:** WOLA requires that each register name within a region is unique. The Data Virtualization Manager server interface to z/OS Connect ensures that the register name is unique within all connection definitions.
WNAME(*WOLA-name*)	This required parameter is used to supply the first part of the WOLA name for the connection to z/OS Connect.	The value may be up to 8 characters long, the first character must be alpha or national, and the rest must be alphanumeric or national. The value is treated as uppercase. The value supplied here must match the 'wolaGroup' parameter of the 'zosLocalAdapters' entry in the z/OS Connect `server.xml` file.

Note: Additional parameters for customizing the connection are described in "Data Virtualization Manager configuration member parameters" on page 36. Contact Technical Support for instructions on how to use these parameters.

Starting the Data Virtualization Manager server and z/OS Connect server

To use z/OS Connect, you need to start the Data Virtualization Manager server and z/OS Connect server. The start up process establishes a connection between the two servers.

Procedure

1. After making the modifications for z/OS Connect, re-start the Data Virtualization Manager server using the following console command:

   ```
   S VDBS
   ```

   Note: If you want to stop the server, issue the following console command: P VDBS
2. To start a z/OS Connect server:

   ```
   S BAQSTRT,JOBNAME=your_jobname,PARMS='<server name> --clean'
   ```

   Note: To stop a z/OS Connect server, use a command like the following, `/p <server name>`.

   After starting the servers, the z/OS Connect server log shows the connection to the Data Virtualization Manager server and indicates that the "Data Integration" Web Service Operations have loaded.

You will see the following messages in the Data Virtualization Manager server log that indicate it is connected to the z/OS Connect server:

```
ZCPHASY subtask is active
ZCPRHUPR subtask is active
ZCPRHLWR subtask is active
```

Results
You are now ready to use the z/OS Connect solution.

z/OS Connect

This section contains detailed z/OS Connect configuration information, including configuring the Data Virtualization Manager server by using the DEFINE ZCPATH command, defining Data Virtualization Manager configuration member parameters, and configuring the Service Provider on zcEE.

Data Virtualization Manager server configuration

The **DEFINE ZCPATH** command can be used to define a connection to a specific z/OS Connect region (server).

DEFINE ZCPATH

Normally, you would place the required **DEFINE ZCPATH** commands in the AVZSIN00 member for a specific Data Virtualization Manager region. Note that the Data Virtualization Manager will (by default) retry a failed connection (caused by, for example, the z/OS Connect server being not active).

Make sure the RNAME matches the `registerName` in your `server.xml` file. Make sure the WAME matches the `wolaGroup` in your `server.xml` file.

Security

When a user is being checked to see if they are allowed to execute the relevant command, the resource name of ZCPATHS is used. The ZCPATHS is checked by the internal rules and no external security is involved.

```
Define ZCPATH
    NAME(defname)
    RNAME(register-name)
    WNAME(WOLA-name)
    [ WNAMEX(WOLA-namex) ]
    [ STATUS({ Active | Inactive } ) ]
    [ RETRY( { NO |interval }) ]
    [ MINCR(n) ]
    [ MAXCR(n) ]
    [ MAXQR(n) ]
    [ MAXCACT({ ERROR | QUEUE} ) ]
    [ MAXPRQ(n) ]
    [ MAXQT(n) ]
    [ USERID(userid) ]
    [ MAXPTIME(n) ]
    [ MAXRECS(n) ]
    [ MAXRLEN(n) ]
    [ USEACTUID( { NO | YES } ] }
```

The required parameters (**NAME**, **RNAME**, **WNAME**) for **DEFINE ZCPATH** are described in "Modifying the Data Virtualization Manager configuration member" on page 30.

The optional parameters for the **DEFINE ZCPATH** command are as follows:

Parameter	Description	Valid values
NAME(*defname*)	This required parameter is used to provide a specific definition name.	A definition name may be up to 20 characters long, the first character must be alpha or national, and the rest must be alphanumeric or national, and the '_' and '.' are also allowed. The value is always treated as uppercase.
RNAME(*register-name*)	This required parameter is used to supply the register name for the connection to z/OS Connect. The name supplied here must match the RegisterName value on the localAdaptersConnectService description in the z/OS Connect `server.xml` file.	A register name may be up to 12 characters long, and the first character must be alpha or national, and the rest must be alphanumeric or national. The '_' and '.' are also allowed. The value is *not* treated as uppercase. **Note:** WOLA requires that each register name within a region is unique. The Data Virtualization Manager server interface to z/OS Connect ensures that the register name is unique within all connection definitions.
WNAME(*WOLA-name*)	This required parameter is used to supply the first part of the WOLA name for the connection to z/OS Connect.	The value may be up to 8 characters long, the first character must be alpha or national, and the rest must be alphanumeric or national. The value is treated as uppercase. The value supplied here must match the 'wolaGroup' parameter of the 'zosLocalAdapters' entry in the z/OS Connect 'server.xml' file.
STATUS({ ACTIVE | INACTIVE }) *(Optional)*	This optional parameter is used to supply the status of the connection.	If omitted, the default is 'ACTIVE'. STATUS=INACTIVE can be used to create a definition, but not attempt to connect to the z/OS Connect region.
RETRY({ NO | interval }) *(Optional)*	This Optional parameter is used to supply an overriding retry interval (in seconds) or prevent retry (by specifying NO).	Integer (in seconds) If this parameter is not specified, then the value of the ZCONNECTPRETRY initialization parameter is used.
MINCR(*n*) *(Optional)*	This optional parameter allows you to indicate the minimum number of worker tasks that will be pre-started or left active to service requests.	The valid range is 1 to 10. If this parameter is not specified, then the value of the ZCONNECTPMINCR initialization parameter is used.

Parameter	Description	Valid values
MAXCR(*n*) *(Optional)*	This optional parameter allows you to indicate the maximum number of worker tasks that can exist at any one time (== concurrent requests).	The valid range is 5 to 20. The value must be equal to or greater than the MINCR value. If this parameter is not specified, then the value of the ZCONNECTPMAXCR initialization parameter is used. See the notes associated with the ZCONNECTPMAXCR initialization parameter.
MAXQR(*n*) *(Optional)*	This optional parameter allows you to indicate the maximum number queued requests (awaiting worker tasks) that can exist at any one time.	The valid range is 0 to 50. If this parameter is not specified, the value of the ZCONNECTPMAXQR initialization parameter is used. See the notes associated with the ZCONNECTPMAXQR initialization parameter.
MAXCACT({ ERROR | QUEUE }) *(Optional)*	This optional parameter allows you to indicate the action to take if a new request arrives while the server is already processing the maximum number of concurrent requests as set by the MAXCR parameter.	ERROR means that an error is returned to the requestor, and QUEUE means that the request will be queued (see the MAXQT parameter). If this parameter is not specified, then the value of the ZCONNECTMAXQACT initialization parameter is used.
MAXPRQ(*n*) *(Optional)*	This optional parameter allows you to set how many requests a given processing task may handle before it is recycled. This can be used to prevent any storage leaks during processing of requests.	The range is 0 to 1000. If the value is 0, no recycling is done. If this parameter is not specified, then the value of the ZCONNECTPMAXPRQ initialization parameter is used.
MAXQT(*n*) *(Optional)*	This optional parameter allows you to set a maximum queue time when MAXCACT is QUEUE.	Integer (in seconds) If the value (in seconds) is exceeded, then the request is failed (as if MAXCACT=ERROR was specified). The range is 0 to 600. If the value is 0, a request is queued indefinitely. If this parameter is not specified, then the value of the ZCONNECTPMAXQT initialization parameter is used.

Parameter	Description	Valid values
USERID(*userid*) *(Optional)*	This optional parameter allows you to nominate a default userid for the request. If unable to determine the userid associated with the request, this value is used.	The value may be up to 8 characters long, and is in 'PDS member name' format – the first character must be alpha or national, and the rest must be alphanumeric or national. The value is treated as uppercase. Note that a value of '*' is also allowed (see below). If this parameter is not specified, the value of the ZCONNECTPUSERID initialization parameter is used. A value of * means that the region userid will be used. **Note:** This userid is the one applied if the value for SNAME is matched with the incoming service name.
MAXPTIME(*n*) *(Optional)*	This optional parameter allows you to set a maximum processing time (in seconds) for a request.	If the time limit is exceeded, an error is returned. The range is 0 to 120. A value of 0 means that no time limit is imposed. If this parameter is not specified, the value of the ZCONNECTPMAXPTIME initialization parameter is used.
MAXRECS(*n*) *(Optional)*	This optional parameter allows you to set a maximum number of returned records for a request. If the value is exceeded, an error is returned.	The range is 1 to 10,000. If this parameter is not specified, then the value of the ZCONNECTPMAXRECS initialization parameter is used.
MAXRLEN(*userid*) *(Optional)*	This optional parameter allows you to set a maximum length of the reply for a request. If the value is exceeded, an error is returned.	The range is 100 to 50,000,000. If this parameter is not specified, then the value of the ZCONNECTPMAXRLEN initialization parameter is used.

Parameter	Description	Valid values
USEACTUID({ NO | YES }) *(Optional)*	This optional parameter allows you to indicate whether the actual userid (if available) is to be used.	NO means that the command-nominated (or defaulted) userid is to be used. YES (The default if this parameter is not specified on the command) means that the actual userid (if available) is to be used. If the actual userid is not available, processing continues as if this operand is NO. **Note:** This USEACTUID value is used if the value for SNAME is matched with the incoming service name.

Data Virtualization Manager configuration member parameters

Except for the ZCONNECT parameter, the value of these z/OS Connect parameters can be changed at any time.

However, while the parameter values can be altered at any time, the latest values of the parameters at the time a **DEFINE ZCPATH** command is processed will be used.

The parameters would normally be specified in the AVZSIN00 configuration member for a Data Virtualization Manager region.

All parameters other than the tracing parameters associated with the z/OS Connect facility are grouped into the PRODZCONNECT group.

Note: Contact Technical Support when using these parameters.

Parameter	Description	Valid values
ZCONNECT	This parameter controls whether the z/OS Connect facility will be available in this Data Virtualization Manager server. If set to NO, then nothing can be done including defining specific connections (They will be ignored with an error message). Note that the value of this parameter cannot be changed after a Data Virtualization Manager server region starts.	**NO** z/OS Connect support is not active. **YES** Activate z/OS Connect support. (default value)
ZCONNECTPWNAMEX	The default second and third parts of the name used to connect via WOLA to the z/OS Connect facility. Each part is 1-8 characters and is separated by a '.' From the next part. A specific connection definition may override this value.	NAME2.NAME3

Parameter	Description	Valid values
ZCONNECTPMINCR	The default minimum number of processing threads that are pre-attached to process incoming requests. A specific connection definition may override this value.	1-30 (default value 2)
ZCONNECTPMAXCR	The default maximum number of processing threads that are allowed. If a request comes in while the server is already processing this number of requests, an error may be returned to the requestor. A specific connection definition may override this value. See the notes below ***1** and ***3**.	5-30 (default value 10)
ZCONNECTPMAXQR	The default maximum number of queued requests allowed if a new request arrives when the path is already processing the maximum number of concurrent requests. A specific connection definition may override this value. See the notes below ***2** and ***3**.	0-50 (default value 10)
ZCONNECTPMAXQACT	The default action to take if the maximum number of queued requests is reached and a new request is received. Values are ERROR (Return an error to the requestor) and QUEUE (queue the request until a processing thread becomes available). A specific path definition may override this value.	ERROR
ZCONNECTPMAXQT	The default time that a queued request (as described in the previous parameter) is allowed to wait for a processing thread. If this time is exceeded, it is if ZCONNECTPMAXQACT=ERROR was in effect.	0-600 (default value 0) **Note:** If the value is 0, the request can wait indefinitely. A specific connection definition may override this value.
ZCONNECTPMAXPRQ	The default maximum number of requests that a given 'processing thread' will handle before it is recycled. This allows any storage leaks to be controlled.	0-1000 (default value 10) **Note:** If the value is 0, no recycling is performed. A specific connection definition may override this value.
ZCONNECTPMAXPTM	The default maximum time allowed to process a request. If a request cannot be completed in this time, an error is returned to the requestor. The value is in seconds. A specific connection definition may override this value.	0-600 (default value 10) **Note:** A value of 0 means no limit.

Parameter	Description	Valid values
ZCONNECTPMAXRECS	The default maximum number of records that can be returned on a (query) request. If the number of records is exceeded, an error is returned to the requestor. A specific connection definition may override this value.	1-10,000 (default value 1000)
ZCONNECTPMAXRLEN	The default limit on the length (in bytes) of the response that can be delivered on a request. If the size (in bytes) of the response would exceed this value, an error is returned to the requestor. A specific connection definition may override this value.	100-50,000,000 (default value 100,000)
ZCONNECTPRETRYINT	The default retry interval, in seconds. If unable to connect to a specific z/OS Connect instance, this parameter sets the number of seconds before the path is retried. A specific connection definition may override this value.	0-600 (default value 30) **Note:** If the value is 0, no retry is performed. If less than 10, 10 is used.
ZCONNECTPUID	If we cannot obtain a suitable userid from the WOLA or the z/OS Connect facility, this parameter can be used to nominate a userid under which the requests will run (for authorization purposes). '*' means that the region userid is used. **Note:** This parameter will not be needed if there is a userid.	NOUIDSET
ZCONNECTTRLEV	Sets a trace level. Customers can use this parameter (under guidance). Level meanings: 0: No tracing 1: Trace path start/stop 2: 1 + trace request begin/end 3: 2 + trace request information 4: 3 + trace i/o buffers **Note:** This parameter this parameter is ***not*** in the 'PRODZCONNECT' parameter group, but is in the tracing parameter group.	0-9

Configuring Service Provider on zcEE

After you create a z/OS Connect server, you need to edit your `server.xml` to configure support for Service Provider on zcEE.

About this task

See the IBM documentation about configuring a z/OS Connect server.

Procedure

After you create a z/OS Connect server, make the following edits to your `server.xml` to include the Service Provider on zcEE.

Note: In the various XML configuration file fragments below, colors and arrows are used to show specific connections that must be done correctly.

- Under `<featureManager>`:

```
<feature>usr:dvsProvider</feature>
<feature>zosLocalAdapters-1.0</feature>
```

  Where `usr:dvsProvider` is the Service Provider on zcEE.

  zosLocalAdapters-1.0 is the WOLA provider used to communicate between the Service Provider on zcEE and the Data Virtualization Manager server.

- Be sure to include the HTTP entry where you indicate the socket ports the server will listen on:

```
<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="12345"
httpsPort="12346" />
```

- Include whatever security related entries are appropriate for your environment. For example:

```
<webAppSecurity allowFailOverToBasicAuth="true" />
<safRegistry id="saf" />
<safAuthorization id="saf2" />
<zosconnect_zosConnectManager globalAdminGroup="ZCONUSER"
              globalOperationsGroup="ZCONUSER" globalInvokeGroup="ZCONUSER"/>
```

  Note: These security entries must have some corresponding RACF entries on z/OS. For example, each user would need to be added to the ZCONUSER group. See the IBM z/OS Connect documentation.

At the top level add these entries for the Service Provider on zcEE:

```
<zosconnect_zosConnectService
        id="zosConnectDvsService"
        serviceName="dvsService"
        serviceRef="dvsService"
        serviceDescription=" Rocket DV Service Provider"
        invokeURI="/dvs"/>
<usr_dvsService
        id="dvsService"
        connectionFactoryRef="wolaCF"
        registerName="DVSR1"
        serviceName="DVSS1"
        invokeURI="/dvs"/>
```

And these entries for the WOLA service provider that DV server connects to:

```
  <zosLocalAdapters
    wolaGroup="NAME1"
    wolaName2="NAME2"
    wolaName3="NAME3" />

  <connectionFactory id="wolaCF" jndiName="eis/ola">
    <properties.ola />
  </connectionFactory>

  <zosconnect_zosConnectService id="sdef1" serviceName="dvs1"
serviceAsyncRequestTimeout="600s" serviceRef="svc1"/>

  <zosconnect_localAdaptersConnectService id="svc1" registerName="DVSR1"
serviceName="DVSS1" connectionFactoryRef="wolaCF" connectionWaitTimeout="7200" />
```

This NAME1 corresponds to the WNAME entry in the DEFINE CPATH in the server IN00 config (see previous section).

Note: The values of the `wolaName2` and `wolaName3` parameters can be any value, but must match the values used by the Data Virtualization Manager region. The values shown (`wolaName2 = 'NAME2'` and `wolaName3 = 'NAME3'`) are the default values used by a Data Virtualization Manager region.

Chapter 6. DFStor support

IBM Data Virtualization Manager for z/OS supports DFstor capabilities by enabling users to insert data from a data frame into the target of your choice.

For example, if you have an IBM DB2 Analytics Accelerator that contains critical business data and you are using Spark to perform real-time analytics, Data Virtualization Manager enables you to create a union from data on the accelerator with analytical data on Spark to create invaluable new insights.

By creating a virtual table for both sources in the Data Virtualization Manager and creating a union of both sets of data, you can easily combine data from both Spark and other data sources to achieve greater insights. For procedures about defining your target table for DFstor to DB2 and to IBM DB2 Analytics Accelerator, refer to the IBM Knowledge Center.

Chapter 7. The Data Virtualization Manager studio

This section presents the information you need to access to your mainframe data using the Data Virtualization Manager studio.

Data Virtualization Manager studio overview

This topic introduces the Data Virtualization Manager studio and the function it provides in IBM Data Virtualization Manager for z/OS.

Data Virtualization Manager 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 Virtualization Manager provides data integration without the complexity and cost associated with extracting, transforming, and loading the data.

Data Virtualization Manager studio is an Eclipse-based user interface that communicates with the Data Virtualization Manager server. You use the studio to create metadata objects, such as virtual source libraries, virtual tables, virtual collections and virtual views on the host Data Virtualization Manager 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.

Getting started with the studio

The following list highlights the steps to start using Data Virtualization Manager studio:

- Before you can begin using the Data Virtualization Manager studio, you must first open the DV Data perspective from the **Window** menu, and then connect to the Data Virtualization Manager server on the mainframe that has the data that you want to access.
- To get access to the mainframe data, use the Data Virtualization Manager studio to create the following components on the Data Virtualization Manager 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**.
  - *Virtual collections*: When you create a virtual table, a virtual collection for NoSQL access to data is automatically created. You generate the JavaScript from the virtual collection and use that JavaScript to read and extract the data from the mainframe. Edits that you make to either the virtual table or collection, are automatically applied to both. To edit the contents of a virtual table, you must edit the SQL virtual table.

    Note: Because the virtual table and virtual collections wizards require that you to enter the same information, only the virtual table wizards are documented.

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.

DV Data perspective

The **DV Data** 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 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 Data Virtualization Manager 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. • **Favorites**: Lists shortcuts to the mainframe resources that you frequently access.

Views	Description
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.
NoSQL for MongoDB results	Use to display the results returned from a query.
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.
NoSQL for MongoDB	Use to edit Mongo JavaScript scripts.
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.
- Creating virtual collections for NoSQL access to data.
- Generating application code from SQL.

Services perspective

The **Services** perspective provides the default views, wizards, and editors that you use to perform tasks that are associated with creating and managing services.

The **Services** perspective allows you to manage the implementation of a Service-Oriented Architecture (SOA) with your mainframe applications. This perspective provides the tools needed to build, develop, and transform mainframe applications into web services and components. With the **Services** perspective, users can easily explore resources on the mainframe, and browse mainframe application

screens, define target systems, create Web Services, and related components, and generate components that capture mainframe application logic.

The following views are available from the **Services** perspective.

Views	Description
Services Navigator View	Provides shortcuts to the following: • **Services** – A shortcut to the **Services** folder on the **Server** tab. • **New Target System** – Starts the **Target System Wizard** that is used to create a target system on the Data Virtualization Manager server. The Web Services operations use the target system to map to your mainframe resources. • **New Web Services Directory** – Starts the **Web Services Directory Wizard** that is used to create a Web Services directory. The Web Services directly identifies the PDS on the mainframe that is used to get the metadata libraries. • **z/OS Connect Configuration** – Starts the **z/OS Connect Configuration Wizard** that is used to create the sample XML fragments for use in a z/OS server.xml file. • Screen Logic Configurator • New Component – the Component Wizard to walk you through the process of creating a component for either SLI or BLI.
Explorer	Displays the following tabs and folders: • **Client** tab (local machine view) – **zOSConnect** folder – Contains the generated SAR files and the server.xml file. – **zServices.Projects** folder – Contains related zServices project files such as Web Services downloads. • **Server** tab – **Web Services** folder – Contains the Web Services and operations that are executed to get the mainframe data. – **Target Systems** folder – Contains the target systems that Web Services operations will use to map to mainframe resources. – WSC

Editors	Description
Editors are used to modify Web Services component details.	• **Target System Editor** • **Virtual Directories Editor** • **Web Services Editor** • **Web Service Operations Editor**

Connecting to the Data Virtualization Manager server

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

Connecting to the Data Virtualization Manager server

Use the Data Virtualization Manager studio to connect to the Data Virtualization Manager server that is running on an instance of z/OS.

Before you begin

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

Procedure

1. From the Data Virtualization Manager studio menu, click **Window** > **Open Perspective** > **DV Data**.
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 Data Virtualization Manager server is deployed.
   - **Port**: Enter the port number that the Data Virtualization Manager 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 Data Virtualization Manager 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 Data Virtualization Manager 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 studio, click **Window** > **Open Perspective** > **DV Data**.
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 Data Virtualization Manager server. The default is 1200.
Userid	Enter the mainframe user ID.
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*.SAVZEXEC(AVZSIN00) 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 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 Virtualization Manager** > **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 studio uses when creating connection definitions, from the **Window** menu select **Preferences**, expand **DV Data** > **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.

Creating server metadata

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

Creating virtual source libraries

Virtual source libraries point to the information that IBM Data Virtualization Manager for z/OS 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 Virtualization Manager (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 Virtualization Manager studio.
- Open the DV Data perspective.
- Connect to the Data Virtualization Manager 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.

Virtual collections for NoSQL data access

When you create a virtual table, a virtual collection for NoSQL access to data is automatically created. You can generate the JavaScript from the virtual collection, and use that script to read and extract the data from the mainframe. Because the virtual table and collections wizards require the same information be entered, only the virtual table wizards are documented. Edits that you make to either the virtual table or collection, are automatically applied to both. To edit the contents of a virtual table, you must edit the SQL virtual table.

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.

Field	Action
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.
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 "Generating and executing SQL queries" on page 81.

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 Virtualization Manager 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 *Installation and Customization Guide*.

Note: The data access method options are not displayed if the Data Virtualization Manager 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.
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 Virtualization Manager 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

Field	Action
	(2) and the discovered value. To disable **MapReduce**, select the **Disable MapReduce** check box.

5. In the results table, review the list of selected entries. Modify the selections as needed.

   Tip: Use the check box in the header row of the table to control the selection of all entries.
6. Click **Finish**.

What to do next
Use the studio to easily compose and execute SQL queries using your new virtual tables. See "Generating and executing SQL queries" on page 81.

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 49.

To use the IMS Direct feature, the `IMSDIRECTENABLED` parameter must be enabled in the Data Virtualization Manager 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 56.

   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 57.

   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 58.

      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 "Generating and executing SQL queries" on page 81.

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)** • **Use IMS-Direct (read-only, high performance bulk data access)**	Select the IMS protocol to use. Choose **Use IMS/DBCTL (read/write, transactional integrity)** to use IMS API calls. 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 Data Virtualization Manager server IN00 file. You must select this option for the DBD to be able to enable IMS Direct for a virtual table.

Field	Action
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 55.

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).

Field	Action
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 55.

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.
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 Data Virtualization Manager 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 55.

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 "Generating and executing SQL queries" on page 81.

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.

Field	Action
Metadata Library	From the drop-down list, select the target library where the virtual table metadata will be stored (for example, *hlq*.USER.MAP). The target libraries are specified in the server's started task JCL.
Description	Enter an optional description.
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.
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.
FCT Name (VSAMCICS only)	Enter the name of the VSAM file that is defined for CICS.
CICS Connection Name (VSAMCICS only)	Enter the name of the CICS connection that is configured on the Server.
Mirror Transaction Name (VSAMCICS only)	Enter the name of the Mirror Transaction that is defined for CICS.
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 "Generating and executing SQL queries" on page 81.

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:

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. 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 "Generating and executing SQL queries" on page 81.

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:

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 **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

Field	Action
	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 "Generating and executing SQL queries" on page 81.

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 Virtualization Manager 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.

Field	Description
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 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 studio creates the virtual tables (the metadata maps) on the server.

What to do next
Use the studio to easily compose and execute SQL queries using your new virtual tables. See "Generating and executing SQL queries" on page 81.

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 Virtualization Manager 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 *Installation and Customization 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.
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 "Generating and executing SQL queries" on page 81.

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 Virtualization Manager 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 *Installation and Customization 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 100.

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 Virtualization Manager 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 "Generating and executing SQL queries" on page 81.

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 Virtualization Manager 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 Virtualization Manager 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 *Installation and Customization Guide*.

To use this wizard, virtual tables should already exist for your data sources. See “Creating virtual tables” on page 50.

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 Virtualization Manager 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: – AVZUDT9N – AVZUDTAN – AVZUDTBN – AVZUDTCN For more information, see "Using Db2 for z/OS to access multiple data source types" in the *Installation and Customization 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.

Field	Action
	• **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 Virtualization Manager 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 75.

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 Virtualization Manager 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 "Generating and executing SQL queries" on page 81.

Generated DDL for UDTFs

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

For each Data Virtualization Manager 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 Virtualization Manager. 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"."AVZS_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 "AVZUDTCN"
 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 Virtualization Manager virtual table referenced by a view in Db2. For more information about these parameters, see “UDTF parameters” on page 77.
- The RETURNS TABLE clause describes the columns in the Data Virtualization Manager 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 Virtualization Manager servers.
- The following DDL elements contain information you can customize at view creation time. Considerations for this information are as follows:
  - CREATE FUNCTION "TSUSER"."AVZS_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 "AVZUDTCN" – The external name will always be in the form AVZUDT*a*N, where *a* 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"."AVZS_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"."AVZS_STAFFVS"
 ('',
  'AVZS...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 99.

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 Virtualization Manager 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 ('AVZS...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 Virtualization Manager 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 Virtualization Manager server. This token is for future use and should be omitted. • *vvvvvvvv* – The virtual table name in the Data Virtualization Manager server.

Parameter	Description
`COLINFO ('7,STAFFVS_KEY_ID,...')`	Describes the column count and optionally the Data Virtualization Manager 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 78

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 Virtualization Manager server group name instead of a Data Virtualization Manager 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 Virtualization Manager 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)')

Option	Description
TRACE()	Instructs the UDTF to send trace information to the Data Virtualization Manager 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 Virtualization Manager 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 Virtualization Manager 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 Virtualization Manager studio, where a subset of the virtual table columns have been selected:

```
-- Description:   Retrieve the result set for AVZS_STAFFVS
-- Tree Location: rs99/46000/SQL/Data/Other Subsystems/DSN1/TSUSER/DB2V/
User-Defined Table Functions/AVZS_STAFFVS
-- Remarks:       DB2V:AVZS...STAFFVS
SELECT "STAFFVS_DATA_NAME","STAFFVS_DATA_DEPT","STAFFVS_DATA_JOB","STAFFVS_DATA_YRS"
   FROM TABLE("TSUSER"."AVZS_STAFFVS"
  ('',
   'AVZS...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"."AVZS_STAFFVS"
  ('WHERE STAFFVS_DATA_YRS > 5',
   'AVZS...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);
```

Validating SQL Statements on an SQL Editor

About this task

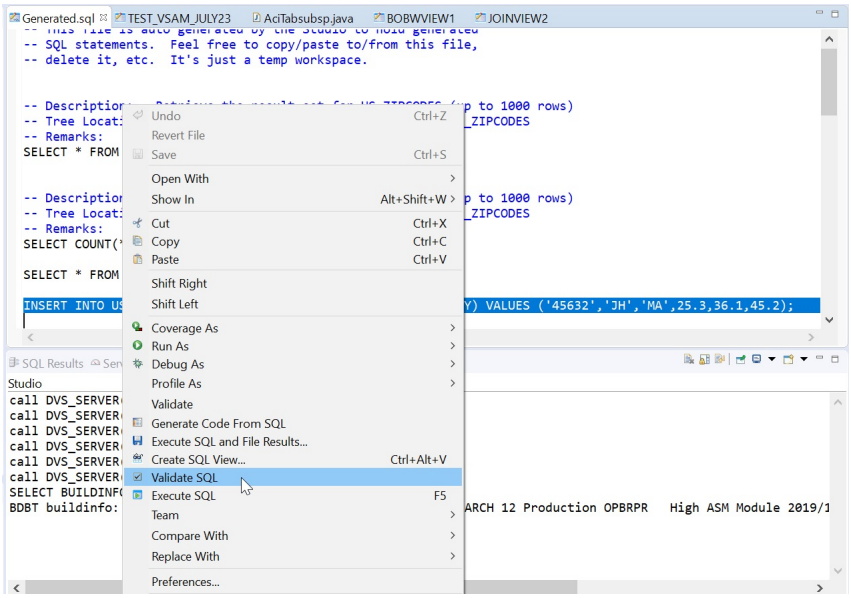
The IBM Data Virtualization Manager for z/OS allows you to validate the following SQL queries that can be run against Virtual Tables and Virtual Views.

- Select
- Insert/Update/Delete
- Queries with parameter markers
- Queries with unnamed column

To validate the SQL queries:

Procedure

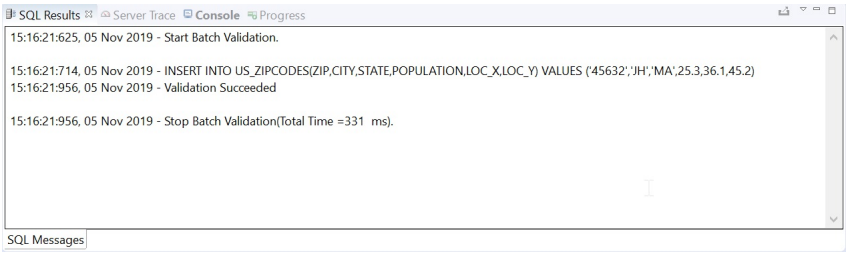
1. Select the query to be validated and right click on it.
2. Select **Validate SQL**.



The system performs the following while validating SQL queries:

- Checks the syntax accuracy.
- Checks if the table name is valid. If it is not valid, it displays the error message **Error: Unable to process map *table_name***.
- Checks if the column name is valid. If it is not valid, it displays the error message **Error: Invalid column reference *column_name* in SQL**.

If the validation is successful, **Validation Succeeded** message is displayed.



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 Virtualization Manager 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 99.

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 Virtualization Manager preferences" on page 96.

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

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.

Field	Action
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 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 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.

Updating the IMS Child Segments

This section describes the conditions when you can insert, update or delete a child segment using the RECORD_ID and the PARENT_ID parameters.

RECORD_ID: This field is the key feedback area from the DLI call for a target segment, and contains the sequence field information for all segments from the root segment to the target segment accessed. Note that if any segment in the path does not have a sequence field, identifying information for those segments is not included.

PARENT_ID: This field is the hex form of the RECORD_ID for a parent segment, and does not include the sequence field of the target segment.

CHILD_ID: This field contains the hex key to the child record if it has a child record.

A child segment can be inserted only if a unique sequence field is present in the child segment's direct parent and all parents in the upward hierarchical path to the root (including the root). If any segment in the hierarchical path does not have a unique sequence field, unambiguously positioning on the direct parent segment for the segment to be inserted is not possible and the segment cannot be inserted.

While inserting a child segment, either the RECORD_ID or the PARENT_ID parameter must be provided as one of the insert values. This column is used to position on the correct parent segment in the database while inserting the new child segment. The RECORD_ID parameter can only be used if all sequence fields up the hierarchical path contain character data (i.e., no integers or packed fields).

The following example shows insertion of child segments using the RECORD_ID parameter.

```
INSERT INTO PART_DI21PART_STAININFO(STANKEY, PROCUREMENT_CODE, MAKE_SPAN, RECORD_ID)
VALUES('13', '09', 100, '02AN960C1 ');
```

The following example shows insertion of child segments using the PARENT_ID parameter.

```
INSERT INTO PART_DI21PART_STAININFO(STANKEY, PROCUREMENT_CODE, MAKE_SPAN, PARENT_ID)
VALUES('13', '09', 100, 'F0F2C1D5F9F6F0C3F1F040404040404040');
```

The RECORD_ID parameter can be used to qualify the segment instance that is to be updated or deleted only if all the sequence fields (i.e., from the root segment down to and including the target segment) are of type character. If not, the PARENT_ID must be used to qualify the parent of the segment and the key of the segment to update must also be provided in the WHERE clause.

The following example shows deletion of elements using the RECORD_ID parameter.

```
DELETE FROM PART_DI21PART_STAININFO WHERE
RECORD_ID = '02AN960C10 08'
```

The following example shows deletion of elements using the PARENT_ID parameter.

```
DELETE FROM PART_DI21PART_STAININFO WHERE
PARENT_ID = 'F0F2C1D5F9F6F0C3F1F040404040404040' AND STANKEY = '08'
```

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 Data Virtualization Manager 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 Virtualization Manager 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 *Installation and Customization 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.

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 AVZMDLDU 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.

Web Services

Use IBM z/OS Connect Enterprise Edition and Data Virtualization Manager Web Services to fully leverage the value of your mainframe data without requiring application developers to be familiar with mainframe systems. Data integration using Web Services makes your mainframe data accessible to new digital technologies through fast, simple, and secure APIs.

When a RESTful API requests mainframe data, z/OS Connect communicates the request to the Data Virtualization Manager server using the WebSphere Optimized Local Adapter (WOLA). The Data Virtualization Manager server executes the requested Web Service to get and return the data to z/OS Connect as objects.

This environment includes the following components:

- Data Virtualization Manager server – Provides mainframe data access to RESTful APIs through z/OS Connect. The Data Virtualization Manager server executes the specified Web Service to get the requested mainframe data and converts and returns that data as objects.
- Data Virtualization Manager studio– Use the **z/OS Connect Configuration Wizard** to generate the Data Virtualization configuration and connection settings that will be included in the z/OS Connect Server.xml file. Use the **Web Services** wizard to create a Web Service and to define the operations required to get the mainframe data.
- z/OS Connect – Enables RESTful API access to existing backend z/OS mainframe services and applications. Routes the data processing requests from a mapped URL of a specific application to the Data Virtualization Manager server.
- WOLA – Provides external access to the internal local communications component of the WebSphere server.

- IBM WebSphere (web server) – Interfaces with remote applications to get data requests over HTTP, REST, and TCP/IP and communicates those requests to the z/OS Connect server.

Getting started

Start the Data Virtualization Manager studio and connect to the Data Virtualization Manager server that will host the Web Services to begin creating Web Services and operations, including:

Setting REST z/OS Connect Web Services preferences

Save your preferred settings to use when invoking and executing a Web Service request for z/OS Connect.

REST via z/OS Connect

To set **Web Services** preferences, complete the following:

1. From the **Window** menu, select **Preferences**.
2. On the **Preferences** page, expand **Data Virtualization** and expand **Web Services**.
3. Select **REST via z/OS Connect**.
4. Complete the following fields:
   - **Service Provider URL(s)** – For each service provider, click **New** and enter the URL of the z/OS Connect server that you want to use to launch REST via z/OS Connect related actions and click **OK**. For example: https://<host>:<port>/12346/dvs. If multiple URLs are entered, the URL of the Data Virtualization Manager server that is currently connected is used.
   - **Max Records Parameter** – Enter the maximum number of records that you want returned. The default value is **20**.
   - **Prompt user before executing generated query** – Enable this option if you want to prompt users before executing a generated query using **REST via z/OS Connect**. By default, after generating a query the query is automatically executed and the generated REST URL is displayed in a web browser.
5. Click **Apply**.
6. Click **OK**.

Connecting to z/OS Connect

Run the **z/OS Connect Configuration Wizard** to create the XML fragments used to connect the Data Virtualization Manager server to your z/OS Connect environment.

Before you begin

If you are using RACF, in order for the z/OS Connect server and the Data Virtualization Manager server to connect and function properly, CBIND resource classes are required. For more information, see "RACF CBIND resource classes" on page 88.

About this task

The **z/OS Connect Configuration Wizard** is used to provide the z/OS Connect systems programmer with the Data Virtualization information to include in the z/OS Connect `server.xml` file. If you do not have all of the information that the wizard prompts you to enter, you can choose to exclude that information from being generated. Completing this wizard is optional.

Procedure

1. To start the **z/OS Connect Configuration Wizard**, in the **Studio Navigator** click **z/OS Connect Configuration**.

2. The **z/OS Connect Configuration** page displays the default IBM Data Virtualization Manager for z/OS service provider settings. You can choose to complete the following Data Virtualization configuration fields (all blank fields are optional):

Section	Fields
HTTP Endpoint	Enter the following Data Virtualization Manager server HTTP endpoint port numbers to use when connecting to a z/OS Connect server: • **HTTP Port** • **HTTPS Port** **Exclude this fragment**: If you already have the HTTP endpoint settings configured in a server.xml file or if you do not know the HTTP port numbers, you can choose to exclude this information from being generated.
WOLA	Enter the names to use to identify a specific z/OS Connect instance or a region in which to connect. To identify a Data Virtualization Manager server on a z/OS Connect instance, enter the following information: • **Register Name**: Identifies a specific path between the Data Virtualization Manager server and a z/OS Connect instance. The name must be unique and it must match the RNAME parameter name that is defined in the Data Virtualization Manager server IN00 file. • **WOLA Group Name**: The name of the WOLA group to use. The name must be unique and it must match the WNAME parameter name that is defined in the Data Virtualization Manager server IN00 file.
Security	**SAF Group**: Enter the z/OS Connect SAF group name. **Exclude this fragment**: If you already have the SAF group name configured in the z/OS Connect `server.xml` file or if you do not know the name, you can choose to exclude this information from being generated.

3. Click **Finish**.

   Review the results that display in the **XML Editor** and make any necessary modifications. You can copy XML fragments from the generated server.xml file into the z/OS Connect server.xml file. You can also choose to export and send the generated server.xml file to the appropriate z/OS Connect systems programmer. The following shows the results of a sample server.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Sample z/OS Connect Server Configuration for the DVS Service Provider.
This is not a complete server.xml file. It contains several XML sections
that you can copy and paste into your actual server.xml file on the host.
-->
<server>
<!--
HTTP Endpoint Details
-->
<httpEndpoint httpsPort="12346" httpPort="12345"
  host="*" id="defaultHttpEndpoint"/>
<!--
Enable SAF Security with Group Name (ZCONUSER)
-->
<webAppSecurity allowFailOverToBasicAuth="true"/>
<safRegistry id="saf"/>
<safAuthorization id="saf2"></safAuthorization>
<zosconnect_zosConnectManager globalInvokeGroup="ZCONUSER"
  globalOperationsGroup="<!--
Adapter Details with WOLA Group Name (RDVWOLA)
-->
<zosLocalAdapters wolaName3="NAME3" wolaName2="NAME2"
  wolaGroup="RDVWOLA"/><!--
        DVS Service Details with Register Name (DVSR1)
    -->
    <zosconnect_zosConnectService invokeURI="/dvs"
    serviceDescription="" serviceRef="dvsService" serviceName="dvsService"
      id="zosConnectDvsService"/>
```

```
    <usr_dvsService invokeURI="/dvs" serviceName="DVSS1"
      registerName="DVSR1" connectionFactoryRef="wolaCF" id="dvsService"/>
    <connectionFactory jndiName="eis/ola" id="wolaCF">
        <properties.ola/>
    </connectionFactory>
    <zosconnect_zosConnectService serviceRef="svc1"
      serviceAsyncRequestTimeout="600s" serviceName="dvs1" id="sdef1"/>
    <zosconnect_localAdaptersConnectService connectionWaitTimeout="7200"
      connectionFactoryRef="wolaCF" serviceName="DVSS1"
      registerName="DVSR1" id="svc1"/>
</server>
```

Note:

Do not use /dvs for a base path name while creating APIs in the IBM Data Studio. This causes a conflict as /dvs is used as the invokeURI attribute for the service provider on zcEE.

What to do next
You can now begin creating the Web Services components using the studio wizards.

RACF CBIND resource classes
If you are using RACF, in order for the z/OS Connect server and the Data Virtualization Manager server to connect and function properly, CBIND resource classes are required.

Either a generic or discrete CBIND resource definition is required, as follows:

- For a generic definition, use a CBIND class of BBG.WOLA.*wolaGroup*.** with UACC(READ).
- For a discrete definition, use a CBIND class of BBG.WOLA.*wolaGroup*.*wolaName2*.*wolaName3* with UACC(READ).

  The default values for *wolaName2* and *wolaName3* are NAME2 and NAME3, respectively, and are defined in the ZCONNECTPWNAMEX parameter. If an existing zcEE server already exists with specified values for *wolaName2* and *wolaName3*, you must change the values in ZCONNECTPWNAMEX to those values.

Creating target systems

Create a target system on the Data Virtualization Manager server that the Web Services operations will use to map to mainframe resources.

Before you begin
Before creating the target system, you must know the name of the code page to use and the type of target system to create.

Procedure

1. On the **Server** tab, expand **Services** > **Target Systems** and click **Create Target System**.
2. On the **Target Systems** page, identify the target system to use by completing the following:
   - **Name** – Enter a unique name for the new target system.
   - **Connection**– Select the SQL92 connection type.
   - **CCSID** – Accept the default mainframe **1047** CCSID or use the drop-down list to select a different mainframe value.
3. Click **Finish**.
   On the **Server** tab, the new target system displays under the **Target Systems** folder.

What to do next
After creating a target system, you can create Web Services directories that are used to identify the PDS on the mainframe where the metadata libraries exist.

Creating Web Services directories

Create Web Services directories to identify the Partitioned Data Set (PDS) on the mainframe where the metadata libraries exist.

Before you begin

Before creating a Web Service directory, you need the following:

- The URL path name that is used to access Web Services.
- The mainframe high-level qualifier to use.
- The name of the microflow library to use. This microflow library is being created for future use.

You must also have the necessary security authorization to perform updates to the metadata libraries.

Procedure

1. On the **Server** tab, expand **Services** > **Web Services** and select **Create Directory**.
2. On the **Web Services Directories** page, complete the following:
   - **Name** – Enter a unique name to identify the Web Services directory on your local machine. You cannot use this name when creating new Web Services within the directory. For example, if you name your directory "*MyWebServices*" you cannot name a Web Service within that directory "*MyWebServices*." This directory is used when generating files that reside on your local machine. No files or file structures will be created on the mainframe.
   - **High Level Qualifier**– Enter a high level qualifier to use as a data set prefix when creating, verifying, or editing the metadata library on the mainframe that is associated with this Web Services directory. By default, the data set name containing the Web Service and operations is `hlq.MMAP`. If this library does not already exist, it will be created.
   - **Supported Protocols**– Select the default **REST via z/OS Connect** protocol which will be used by the Web Services in this directory.
   - **Advanced**
     - **URL Path** – To explicitly define the mainframe metadata library, enter the URL name in the text box. This is a data set where the metadata for all components contained within the library will be stored. For example, `/ZCEE/`.
     - **Metadata Library** – Accept the default metadata library that displays or enter the name of a specific metadata library. For example, *hlq.servername*.MMAP.
3. Click **Next**.
4. On the **Microflow Library** page, under the list of **Current Microflow Libraries** select a library to use or click **Create New Microflow Library** and complete the following fields:
   - **Library Name** – Enter a unique library name.
   - **Library Dataset** – Enter the data set name to use.
5. Click **Finish**.

   The new Web Services directory displays on the **Server** tab under the **Web Services** folder.

What to do next

You can now begin creating Web Services and operations.

Creating Web Services and operations

Create the Web Services and operations that are used to get the RESTful API requested mainframe data and to transform the data into objects.

Before you begin

Before creating a Web Service, you need to know the name of the virtual table or virtual view that will be used to generate the SQL statement. You must also identify or verify the Data Virtualization Manager

server service provider URLs to use to launch REST via z/OS Connect related actions (for details, see "Setting REST z/OS Connect Web Services preferences" on page 86).

Procedure

1. On the **Server** tab, expand **Services** > **Web Services**.
2. Expand the **Web Services** directory where you want the Web Service to reside and click **Create Service**.
3. On the **Web Services** page, enter a unique name for the new Web Service.
   - **Target Namespace**: Accept the default entry, select a different target namespace from the drop-down list, or enter a new target namespace. The default values include; `HTTP://HOST:PORT/<VIRTUAL DIRECTORY PATH>/<WEB SERVICE NAME>`. Where PORT is the services port that the Data Virtualization Manager server should use. This is not the general Data Virtualization Manager server port number. You can choose to modify the default value set in the **Services** preferences.
   - **Restrict IP**: To restrict certain IP addresses from accessing the Web Services, click **Add** and enter the IP address entry.
   - **SSL Support Configuration**: Choose one of the following SSL options.
     - **Mandatory**: Secure connections are required and requests are processed using SSL. If selected, you can also choose to use SSL when retrieving the WSDL for the Web Service by enabling the **SSL to retrieve WSDL** option.
     - **Supported**: Secure connections are supported, but they are not required.
     - **Not Supported**: Secure connections are not supported and they will not be processed using SSL. This is the default setting.
4. Click **Next**.
5. To add a new operation to the Web Service, on the **Web Service Operation Type** page, select **Data Integration (REST via z/OS Connect or SOAP)** and click **Next**.
   - **Data Integration (REST via z/OS Connect or SOAP)**. This is the default setting.
   - **Business Logic Integration (SOAP)**
   - **Screen Logic Integration (SOAP)**
6. Navigate to the virtual table or virtual view to use when generating the SQL statement and click **Next**.
7. On the SQL page, accept the default **Name** and **Description** fields or enter a new name or description.
8. In the **SQL Statement** text box, accept or modify the SQL statement that displays and click **Next**.
9. On the **SQL Results Set** page, verify that the results are correct and make any necessary modifications.
10. Click **Finish**.

    The Web Service operation details display in the **Web Service Operation Editor**. If necessary, modify the details in the editor.
11. To validate your Web Service complete the following:
    - Right-click the operation in your new Web Service and select **z/OS Connect REST Interface** > **Refresh** to synchronize your new Web Service with the service provider.
    - Right-click the operation in your new Web Service and select **z/OS Connect REST Interface** > **Execute Query** to test the new Web Service in a web browser.
12. Optional: To test the Web Service, right-click the new operation or the new Web Service and click **Open in Tester**. Choose one of the following test options:
    - **Default**: use the default test settings.
    - **Create New**: create a new test group that you can name and configure.
    - Select an existing test group.

13. Optional: To create a .sar file, on the **Server** tab, right-click on the Web Service operation and click **Create z/OS Connect SAR File(s)**. The file is saved to your workspace and is available from the **zOSConnect** folder on the **Client** tab.

What to do next
To continue adding more operations under this Web Service group, click **Create New Operation** and repeat the steps starting at step 5.

Output Format

The returned JSON string is in the following format. Note that the order of the fields may be different as z/OS Connect reorders the JSON:

```
{
 “Result”:<result-code>,
 “ErrMsg”:”<error-msg>”,
 “NumRecs”:<n>,
 “Skipped”:<n>,
 “Affected”:<n>,
 “TotRecs”:<n>,
 “NumFields”:<n>
 “Records”:[
  {<record-1-fields],
  {<record-2-fields],
  …
  ]
}
```

Table 3. Syntax Description

Field	**Description**
Result	0 - Request is successful. *Positive Number* - Warning. *Number less than 0 or Negative number* - Error
ErrMsg	Contains the error message returned from the server.
NumRecs	Indicates the number of records returned. If the request is DELETE, UPDATE, or INSERT request, this field will be 0.
Skipped	Indicates the number of returned records that were skipped due to a non-zero **`Skip`** value in the original request. Note that this field will always be present (even for special operations), but will be 0 if no skipping is requested or no records were skipped.
Affected	Indicates the number of records affected by the request. For a DELETE, UPDATE, or INSERT request, this field returns the number of records deleted, updated, or inserted. If the request does not alter any record, this field will be 0. Note that this field will always be present (even for special operations).

Table 3. Syntax Description (continued)	
Field	**Description**
TotRecs	If the original request had a non-zero Skip value or a non-zero N Records value, or if the number of found records exceeded the defined limit (as set by the startup parameter or path definition), this field shows the total number of records found including the number that was not returned due to those limits. If the request is DELETE, UPDATE, or INSERT request, this field will be 0. Note that this field will always be present (even for special operations).
NumFields	Indicates how many fields are in each returned record. Normally, it will be greater than 0. If the request is DELETE, UPDATE, or INSERT request, this field will be 0. If a record has no fields defined to be returned, the field will be 0. In this case, each returned record would just be **`{}`**. Note that this is valid (if unusual) JSON).
Records	This field will be present in the return string if **`NumRecs`** is greater than 0. It is an array containing information about each returned record. The actual fields in each entry depend on the definition of the return data.

Server Trace

Use the **Server Trace** view to record and view Data Virtualization Manager 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 studio calls in the Server Trace results

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

Before you begin

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

Procedure

1. From the **Window** menu, select **Open Preferences** > **DV Data**.
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 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.

Field	Action
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 Virtualization Manager server records in the **Server Trace** view.

Before you begin

Before running **Server Trace**, you must be able to connect to the Data Virtualization Manager 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 Virtualization Manager 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.

Field	Action
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**.

DV Data preferences

Preferences allow you to customize several IBM Data Virtualization Manager for z/OS settings.

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

Data Virtualization Manager preferences

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

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

Field	Description
Enable Server Tracing of Studio Calls	Includes the studio trace calls in your server trace results. This setting is disabled by default.
Studio HTTP Debug Option	The 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 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 (,).

Field	Description
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 studio operation is unsuccessful. The default setting is **30**.
Studio Remote Control Port	The port number that the 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*.SAVZEXEC(AVZSIN00) file: `"MODIFY PARM NAME(PASSWORDCASE) VALUE(ASIS)"`

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 Data Virtualization Manager 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 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 ( ).

Field	Description
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 Virtualization Manager 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.

Field	Description
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.
Connection Overrides	To override the connection settings that the Data Virtualization Manager 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 ( ).

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`.
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.

Field	Description
Use prepared statement to retrieve SQL column info for DB2 or DRDA tables	The Data Virtualization Manager 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 Virtualization Manager 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 Virtualization Manager 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 Virtualization Manager 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.

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 Virtualization Manager 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>`

Field	Description
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 Virtualization Manager 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.

SSL preferences

Use **SSL** preferences to secure JDBC and HTTP network communications between the Data Virtualization Manager studio and the Data Virtualization Manager 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 Virtualization Manager studio and the Data Virtualization Manager server. If enabled, select the **Protocol** version to use for communications between the Data Virtualization Manager studio and the Data Virtualization Manager server. The default setting is **TLS 1.2**.
Server Authentication	Select the authentication strategy to use: • **Require Server Validation**: Enable to require that all Data Virtualization Manager 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 Data Virtualization Manager 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 Data Virtualization Manager 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 Data Virtualization Manager server certificates. If enabled, the Data Virtualization Manager studio does not validate the server certificate. The default setting is **Require Server Validation**.

Field	Description
Client Authentication	To enable client authentication by the Data Virtualization Manager 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.

Creating RESTful Services for DB2 Objects

The Data Virtualization Manager server allows you to create RESTful services for DB2 objects.

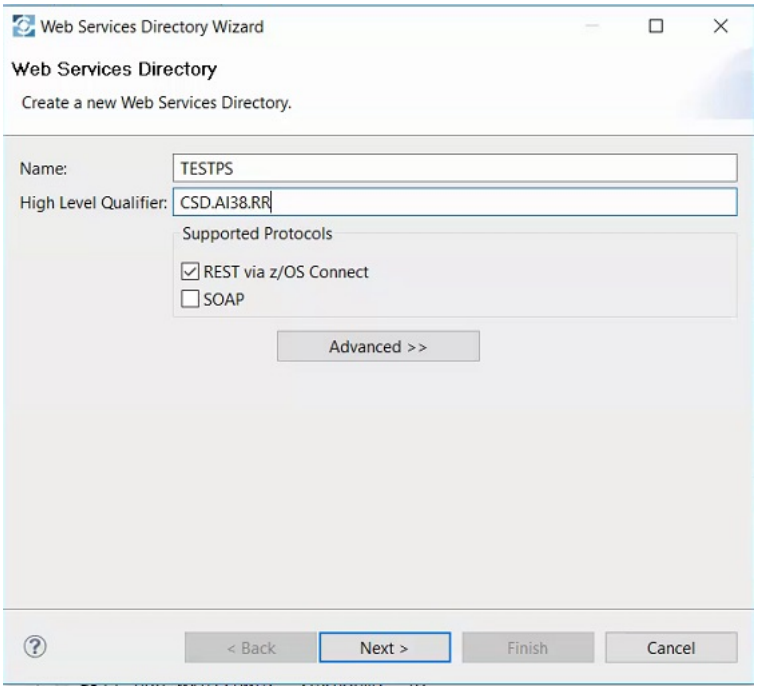
You can create RESTful services for DB2 objects to access the created user-defined table functions (UDTFs). This helps to process the WHERE predicates at DB2 and ensure quick processing of queries.

Creating a Web Service for DB2 Objects

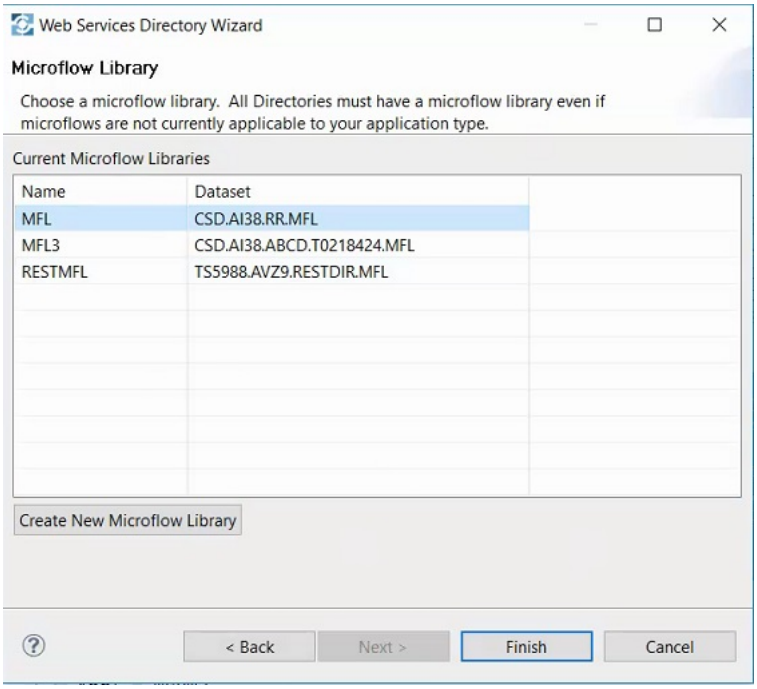
This section helps you to create a new directory and a web service.

Procedure

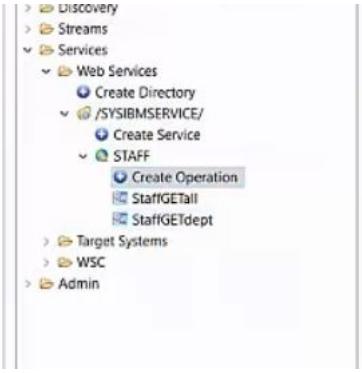
1. Click **Services->Web Services->Create Directory** to create a new directory.



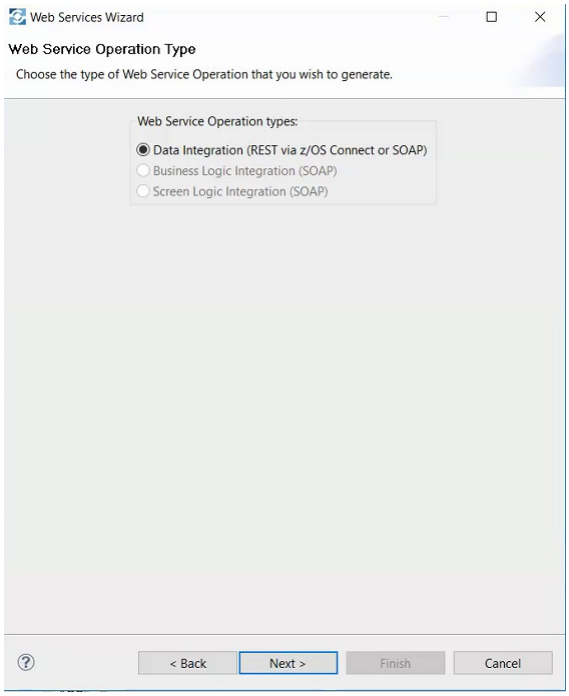
2. Provide a name and the high level qualifier for your server and click **Next**.



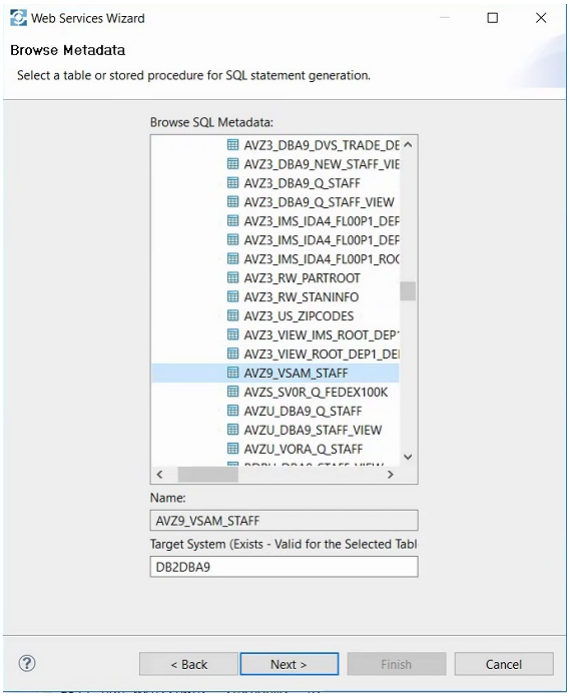
3. Select a microflow library and click **Finish**.
4. From the created directory, click **Create Service.**.



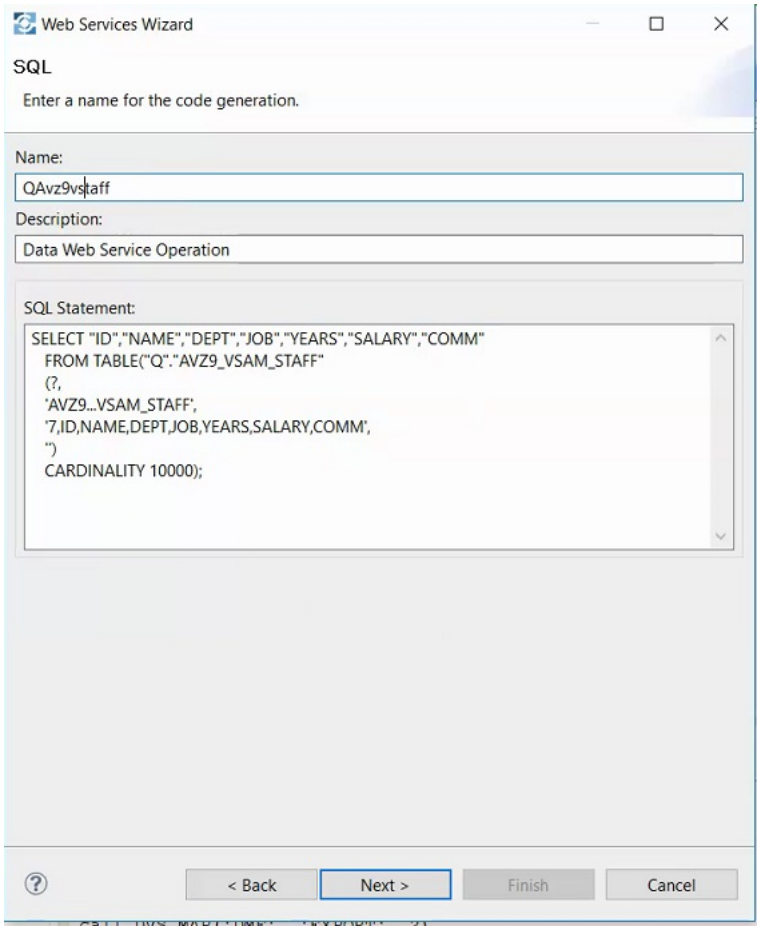
5. provide a name for the service and click **Next**. The following wizard appears.



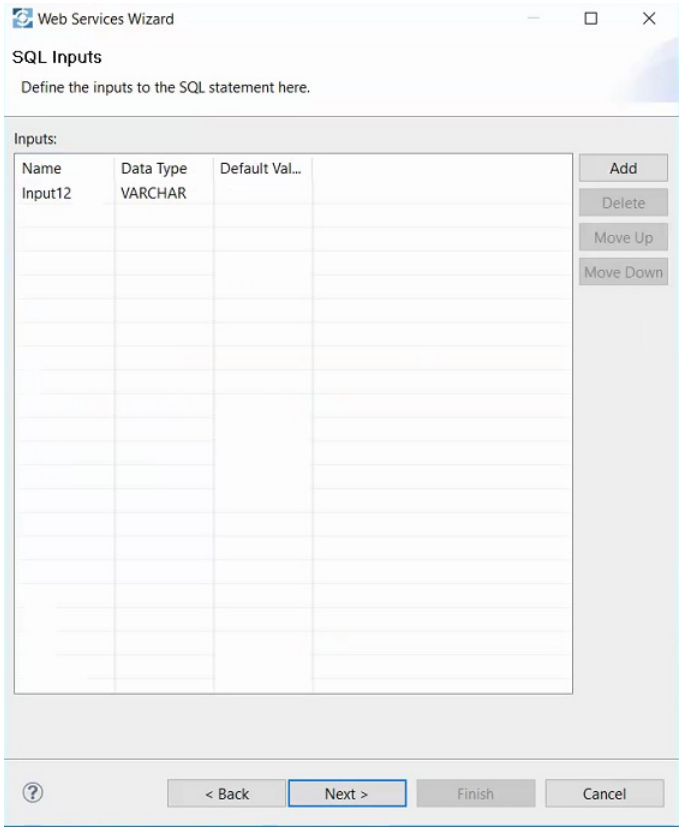
6. Choose **Data Integration (REST via z/OS Connect or SOAP)** and click **Next**.
7. Navigate to your subsystem under **Data->Other Subsystems** and select the UDTF for which you want to create web services in the **Browse Metadata** window.



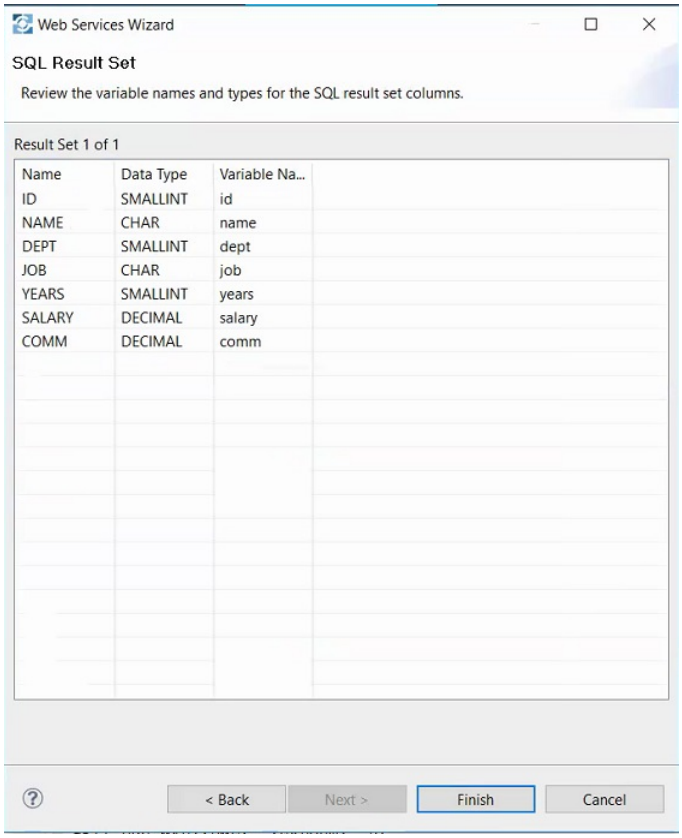
8. Provide a suitable name for the service in the **Name** text box.



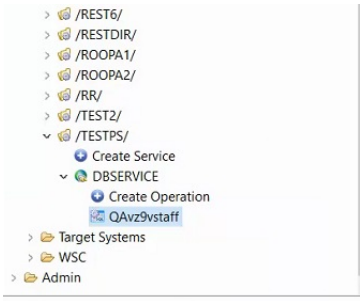
9. Click **Next**.



10. Click **Next**.



11. Click **Finish** to create the operation. The web services is created under the new directory.

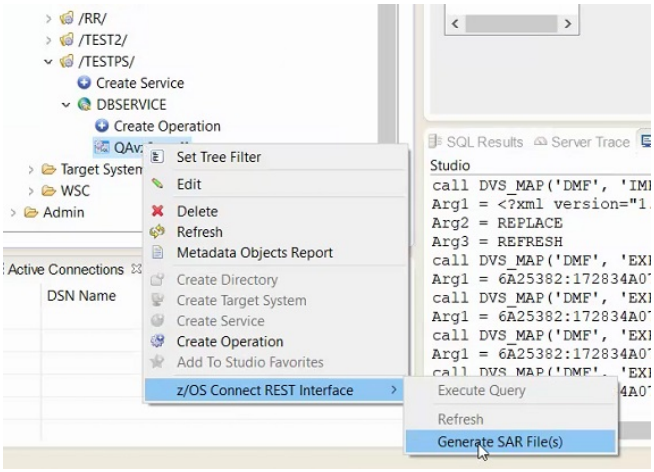


Creating a SAR File

Once the web service is created for a created user-defined table functions (UDTF), you can generate a SAR file.

Procedure

Right click on the created web service and choose **z/OS Connect REST Interface → Generate SAR File(s)**.



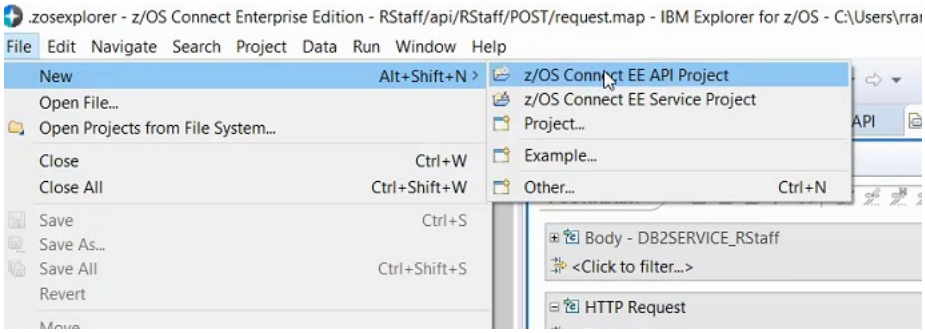
The SAR file is created and the location of the file is shown on the screen.

Creating an API Project

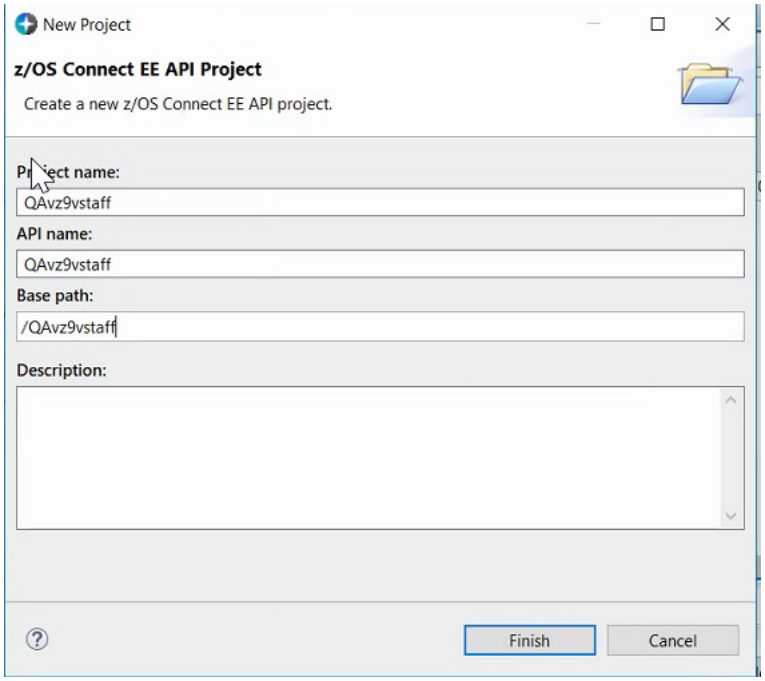
Once the SAR file is created, use it to create an API project in the IBM Explorer.

Procedure

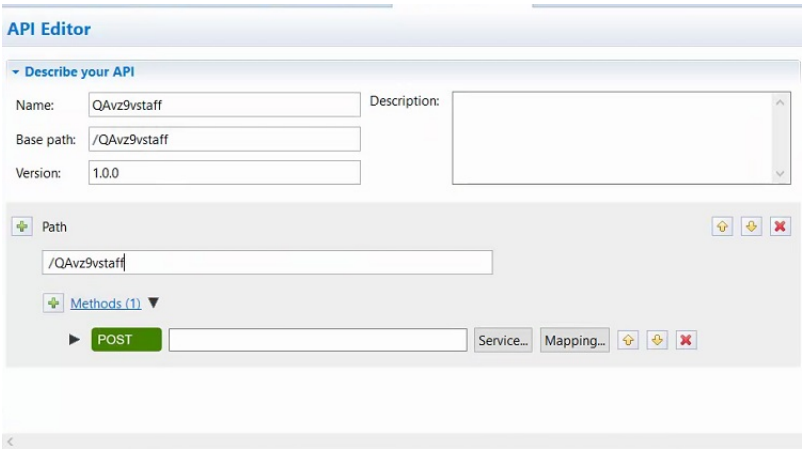
1. Open IBM Explorer and create a new API project.



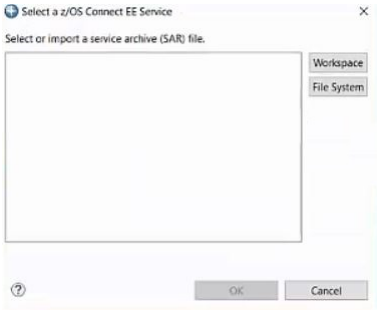
2. Provide **Project name**, **API name**, **Base path**, and **Description** in the new project creation window.



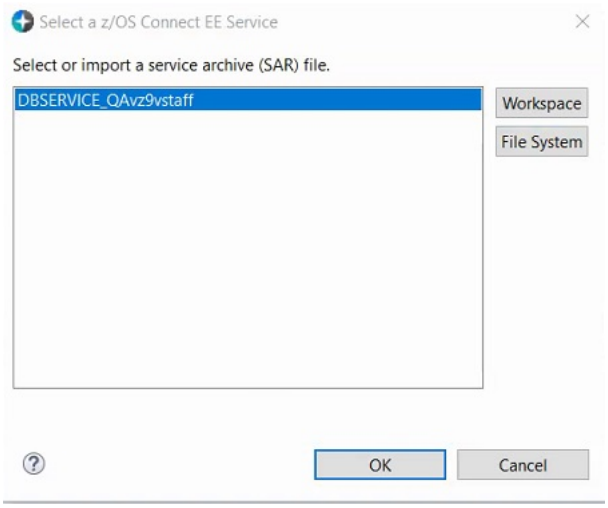
3. Click **Finish**.
4. Provide the path name followed by the query parameter separated by **?**.



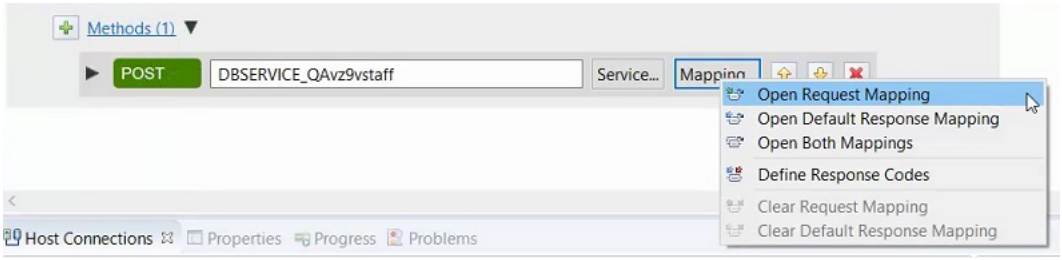
5. Click **Service..** under **POST**.



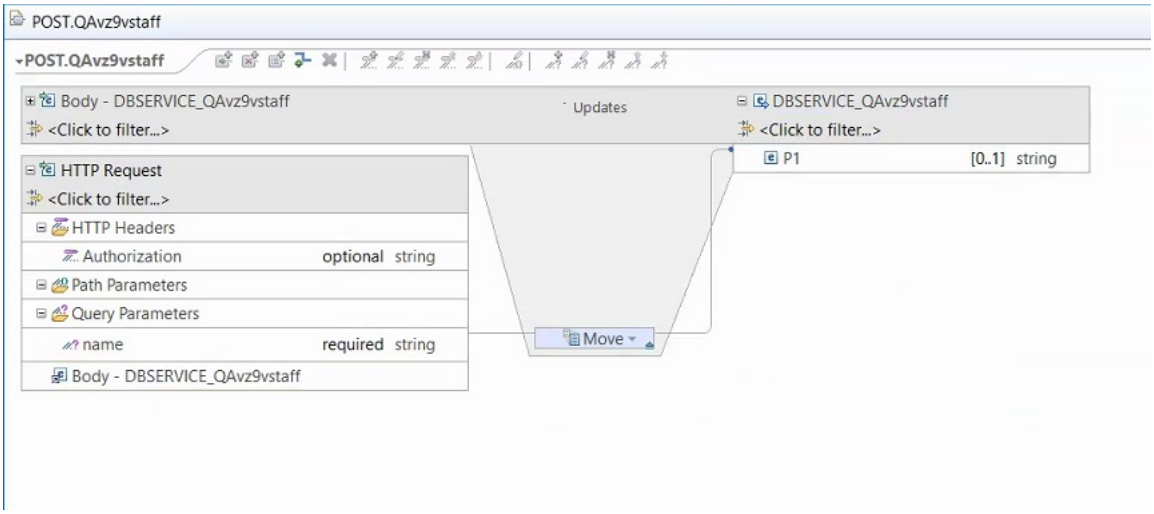
6. Click **File System** and choose the created SAR file.



7. Click **Mapping.. → Open Request Mapping**.



8. Map the created service with the query parameter.

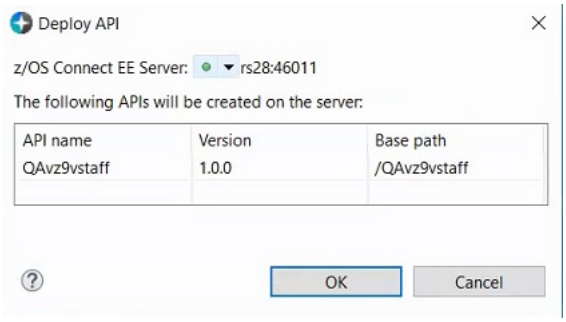


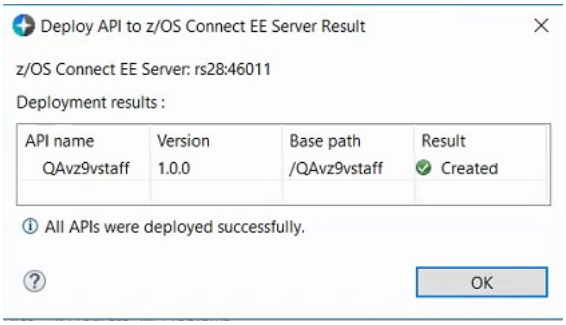
Deploying the Web Service

Once the created service is mapped with the query parameter, deploy the created service.

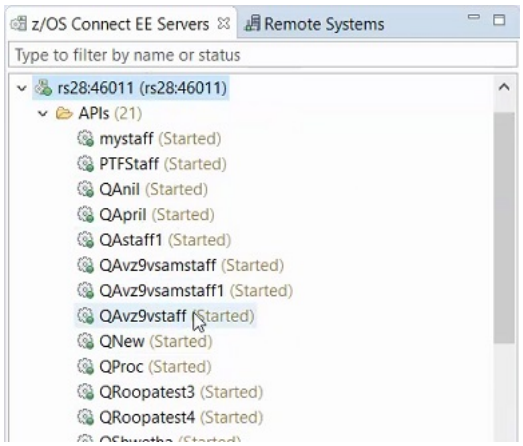
Procedure

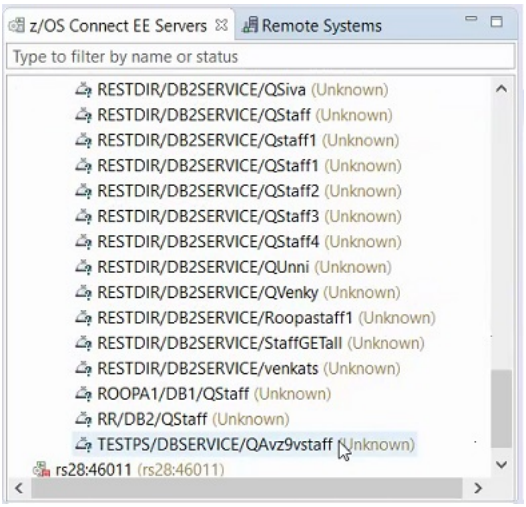
1. Click **File → z/OS Connect EE → Deploy API to z/OS Connect EE Server**.





2. Click **OK**. The deployed service appears.



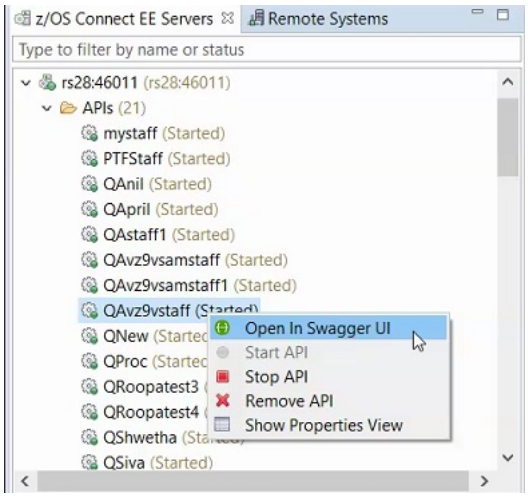


Executing the Web Service

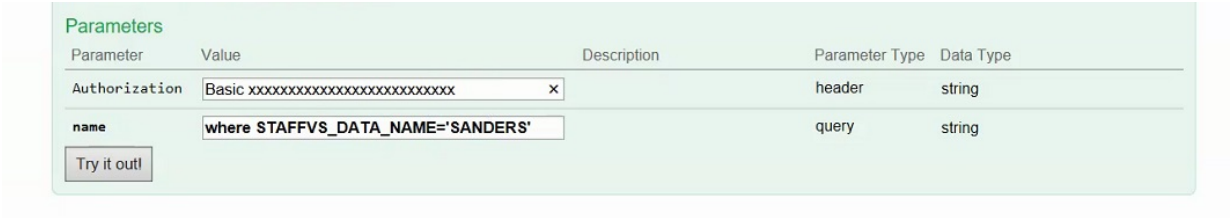
Once the service is deployed, you can start using it.

Procedure

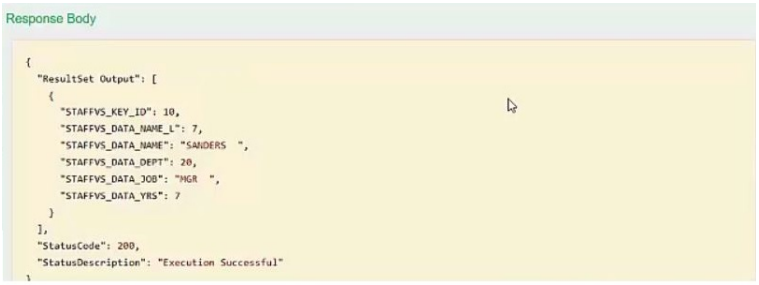
1. On the IBM Explorer, right click on the created web service and click **Open in Swagger UI**.



2. Click **List Operations**.
3. Provide **Authorization** and passing parameter in **name**.



4. Click **Try it out!**. The results are displayed.



Index

R

S

T

U

V

W

Z

SC27-9301-00