THE POWER TO KNOW.

SAS® Federation Server 3.2
Administrator's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® Federation Server 3.2: Administrator's Guide*. Cary, NC: SAS Institute Inc.

SAS® Federation Server 3.2: Administrator's Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

July 2013

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit **support.sas.com/bookstore** or call 1-800-727-3228.

Contents

What's New in SAS Federation Server 3.2

Overview

DataFlux Federation Server is now SAS® Federation Server. This renaming is a result of the integration of DataFlux products into the SAS suite of data quality, data integration, data governance, and master data management solutions.

SAS Federation Server 3.2 is now available through SAS delivery channels. For information about deploying SAS Federation Server, see your SAS Software Order E-mail (SOE).

For product documentation, see the following:

- Documentation for SAS Federation Server is available at:

  http://support.sas.com/documentation/

- The SAS FedSQL Reference Guide replaces the DataFlux FedSQL Reference Guide. The SAS FedSQL Reference Guide is available at:

  http://support.sas.com/documentation/

- SAS Installation Center for System Requirements:

  http://support.sas.com/installation_documentation

The following new features have been implemented in SAS Federation Server 3.2:

- In-Memory Data Store (MDS)
- Privilege Caching

In-Memory Data Store (MDS)

MDS is a transactional in-memory data store that can be implemented on SAS Federation Server. See "Memory Data Store" on page 103 for additional information.

Privilege Caching

Privilege caching is offered as a tool for streamlining privilege determination resulting in improved performance. See “Privilege Caching” on page 77 for additional information.

Chapter 1

Overview

Overview

About SAS Federation Server

SAS Federation Server is a data server that provides scalable, threaded, multi-user, and standards-based data access technology in order to process and seamlessly integrate data from multiple data services. The server acts as a hub that provides clients with data by accessing, managing, and sharing SAS data as well as several popular relational databases.

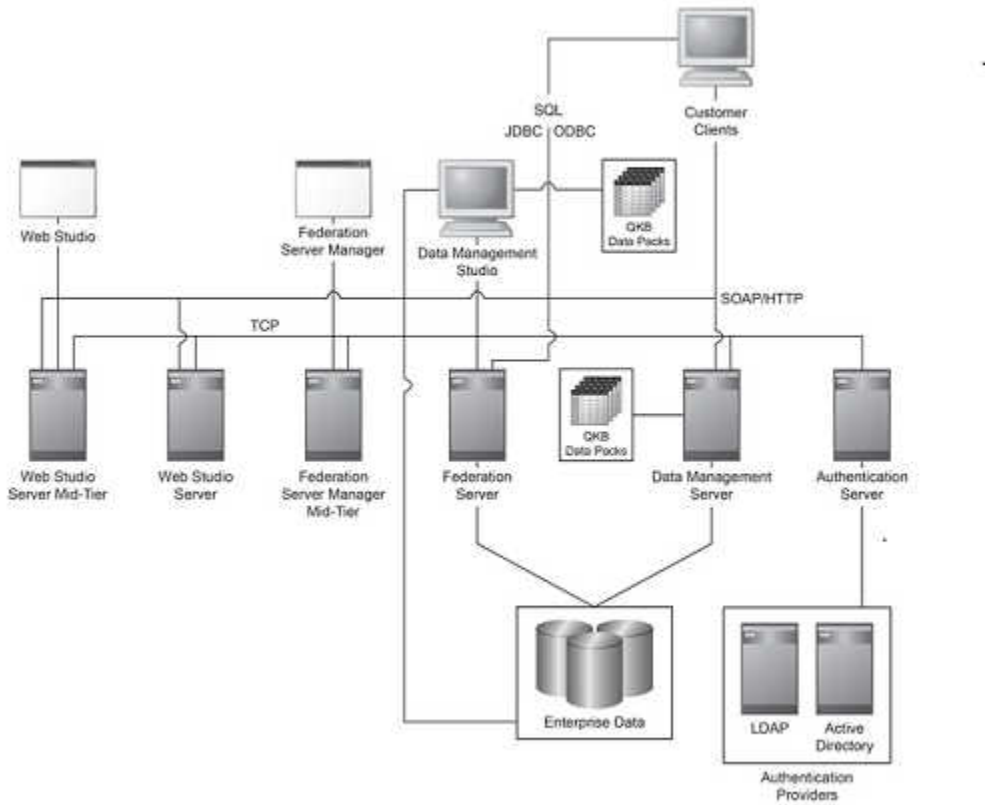
SAS Federation Server enables powerful querying capabilities, as well as improved data source management. With SAS Federation Server, you can efficiently unite data from many sources, without moving or copying the data.

SAS Federation Server provides the following data capabilities:

- A central location for setup and maintenance of database connections.
- Access to popular database systems including DB2, Oracle, SAP, SQL Server, Teradata, and Greenplum.
- ODBC and native drivers to connect to select data sources.
- Threaded data access technology that enhances enterprise intelligence and analytical processes.
- Multi-user services that enable multiple clients to access the same data concurrently.
- Ability to reference data from disparate data sources with a single query, known as data federation. also includes its own SQL syntax, Federation Server SQL (FedSQL), to provide consistent functionality – independent of the underlying data source.
- A data abstraction layer, providing the ability to present a consistent data model throughout the organization. This is accomplished through the use of FedSQL views.
- Data access control with user permissions and data source security.

Integration with SAS DataFlux Data Management Studio and Web Studio

SAS Federation Server is a product that can be used by DataFlux Data Management Studio and Data Management Server. The following figure illustrates the DataFlux Data Management Platform architecture, with SAS Federation Server as a single point of access to provide multiple client applications with data from different sources:



Services Provided by SAS Federation Server

Data Access Technology

The data access technology that is provided by SAS Federation Server consists of a set of run time components that provide a scalable, threaded, multi-user, and standards-based way to process and seamlessly integrate data from multiple data sources. The components provide the data access services that are required by business intelligence and analytical processes.

Threaded Services

Threads are an integral part of a high-performance, scalable system, and they are one of the main features of SAS Federation Server data access technology. Most threaded functionality can be further boosted in an environment in which multiple processors work in parallel. However, performance boosts can also be obtained with multi-threaded processes on a single processor machine.

A threaded service is a method of processing that divides a large job into several smaller jobs that can be executed in parallel. Threaded services control and execute requests by using multiple threads to increase data throughput. A thread is a single path of execution of a process in a single CPU. A thread can also be thought of as a basic unit of program execution in a thread-enabled operating environment. In a symmetric multiprocessing (SMP) environment, which uses multiple CPUs, multiple threads can be spawned and processed simultaneously. Regardless of whether there is one CPU or many, each thread is an independent flow of control that is scheduled by the operating system.

SAS® Federation Server provides threaded services that execute multiple user requests in parallel. Here are examples:

- Each connection to the SAS® Federation Server is managed on a separate thread. This enables multiple users to execute requests in parallel and reduces the probability of a user request being blocked while other user requests are processed.
- Complex requests (or large individual requests) are separated into units of work that are then executed in parallel. For example, filtering operations that require scanning large tables can be processed in parallel, and operations such as sorting can be processed by dividing the result set into subsets, sorting each subset in parallel, and then merging the sorted subsets into the final result set.
- Threading is also used to return result sets on multiple threads. For example, the FedSQL processor can request result sets from disparate data sources on separate threads. By reading data simultaneously, the FedSQL processor can acquire the data faster and expedite results to the client.

In addition to threaded services, some data services provide threaded I/O, which further enhance performance.

Multi-User Services

Multi-user services enable multiple clients to access the same data concurrently. If the data source supports this capability, SAS Federation Server enables two or more clients to write to the same table at the same time without destroying or losing updates. This process is referred to as concurrent update access.

SAS Federation Server uses Integrated Object Model (IOM) technology. IOM technology is a set of object-based interfaces to features or services. The technology enables application developers to use industry-standard programming languages, programming tools, and communication protocols to develop client programs that access these services on IOM servers.

A multi-user environment automatically ensures data protection during concurrent updates. The data services support concurrent updates by locking the data that is being updated and releasing the lock when updates are complete. This prevents loss of data or loss of updates that are due to simultaneous updates.

Performance

SAS Federation Server integrates both user scalability and processing scalability to provide increased performance.

SAS Federation Server supports the following performance capabilities:

- In a multi-user environment, the server automatically scales to the number of concurrent users.
- The server provides rapid access to large amounts of data.
- The server is available under many 64-bit operating environments, which enables the server to scale in-memory processes.
- The server provides application-based, high-performance data reading by supporting a variety of cursor types, multi-row fetch capabilities, and positioned update of result sets.

Data Storage Support

SAS Federation Server provides access to several types of data, which removes the need for an application to connect to one server for some data and another server for other data. SAS® Federation Server supports SAS® data sets, SAP®, and third-party relational databases as data sources.

Relational databases include:

- IBM® DB2
- Oracle®
- Microsoft® SQL Server®
- Teradata®
- Greenplum®

By supporting several data sources, SAS Federation Server gives you the flexibility to configure data storage based on specific needs. You choose the type of data storage that is most appropriate for the particular needs of an application, based on functionality that is provided by each data sources.

For more information about the supported data sources and the functionality that they provide, see the SAS Federation Server Driver Reference on page 110 .

Standards-Based Interface for SQL

SAS Federation Server provides a standards-based interface for the SQL, which defines the data access model for the server. That is, an application creates, requests, and manipulates data by submitting SQL statements.

An application can submit SQL statements by using JDBC and ODBC drivers. The SQL is interpreted by the FedSQL processor, which supports a standard dialect across all back-end data sources. For more information, see "Components of SAS Federation Server" on page 5.

Security

SAS Federation Server security services ensure that both the server and its data are protected against unauthorized access. SAS Federation Server supports configurable authorization processes and other security features, including encryption. In addition, SAS Federation Server provides the ability to control access to SAS data sets that are placed under exclusive control of the server.

For more information, see See "Authentication" on page 68 .

Components of SAS Federation Server

Introduction

SAS Federation Server consists of a set of components that provide the functionality that is required by data integration, business intelligence, and analytic processing. For example, SAS Federation Server provides several types of drivers and two interfaces that you use to connect to the server. The following topics describe each component.

Federation Server Drivers

A SAS Federation Server driver interacts with a data source to read and write proprietary file formats. Each supported data source has a driver that communicates with the data service in its own language to resolve data access requests, and to manage physical files and database tables. For example, to process an SAP table, an application uses the Driver for SAP, and to process an Oracle table, an application uses the Driver for Oracle.

A Federation Server driver provides connectivity to and from the data source, submits SQL statements to the data source, and sends data to and from the data source. That is, a Federation Server driver receives an SQL expression as input and returns the result as output. Each Federation Server driver supports the database functionality of the underlying data source.

For more information about data sources and SAS Federation Server drivers, see "Overview" on page 86.

Language Driver

Language drivers implement the SAS® Federation Server languages by processing a request and sending the parsed query to the appropriate Federation Server driver that

satisfies the request and returns the result. The multi-threaded languages provide a powerful way to create and query data.

FedSQL

FedSQL is the implementation of SQL that SAS Federation Server uses to access relational data. FedSQL is designed to be ANSI SQL:1999 core compliant with some extensions.

For applications, FedSQL provides a common SQL syntax across all data sources. That is, FedSQL is a vendor-neutral SQL dialect that accesses data from various data sources without requiring the application to submit queries in the SQL dialect that is native to the data source. In addition, a single FedSQL query can target data in several data sources and can return a single result set. When possible, FedSQL queries are optimized with multi-threaded algorithms to resolve large-scale operations.

The FedSQL language driver parses FedSQL requests, and then sends the parsed query to the appropriate Federation Server driver to determine whether the functionality can be handled by the data source.

For the SAS Federation Server Driver for Base SAS, the parsed FedSQL request is interpreted 'as is' because FedSQL is the native SQL dialect for the BASE data service.

For the relational database drivers, if the data source supports the functionality, then the FedSQL request is translated to the data source's native SQL dialect. By enabling as much functionality as possible to be handled by the data source, performance is enhanced. However, if a data source does not support the requested functionality, then the FedSQL language driver attempts to compensate by completing the balance of the operation.

For complete FedSQL statement reference, see the*SAS FedSQL Reference Guide.*

Supported Data Sources

Overview

SAS Federation Server supports disparate data sources by providing software in the form of Federation Server drivers, which access the physical data that an application processes. Remote Federation Server drivers provide access to third-party relational databases, such as DB2®, Greenplum®, ODBC data sources, Oracle®, and Teradata®, by connecting to a remote server process.

SAS Federation Server supports the following data sources: SAS data set, SAP®, and several third-party relational databases.

The following table lists the data sources with their supported data service types:

Data Sources	Data Service Types	Default Drivers	Supported Drivers
SAS data sets	BASE	BASE	BASE
DB2	DB2UNXPC	DB2	ODBC, DB2
Greenplum	Greenplum	Greenplum	ODBC, Greenplum

Data Sources	Data Service Types	Default Drivers	Supported Drivers
Oracle	ORACLE	ORACLE	ODBC, ORACLE
SAP	SAP	SAP	SAP
SQL Server	SQLSERVER or SQLSVR	ODBC	ODBC
Teradata	TERADATA	TERADATA	ODBC, TERADATA
ODBC_FED	Native Catalogs	ODBC	ODBC
ODBC	Logical Catalogs	ODBC	ODBC
Memory Data Store	MDS	MDS	MDS

SAS Data Set

The SAS data set is the Base SAS proprietary file format for SAS software, which contains data values that are organized as a table of observations (rows) and variables (columns). The supported file format is the same as SAS data sets that are created by the BASE engine for Version 7 and later. A supported SAS data set uses the extension .sas7bdat.

The SAS Federation Server Driver for Base SAS provides Read and Update access to legacy SAS data sets. In addition, the driver creates SAS data sets that can be accessed by both SAS Federation Server and Base SAS software. The driver supports standard Base SAS storage functionality such as indexing, general integrity constraints, and SAS formats and informats. For more information about supported functionality and compatibility guidelines, See “SAS Data Set Reference” on page 151 .

SAP

For information about supported functionality and compatibility guidelines, See “SAP Reference ” on page 137 .

Third-Party Relational Databases

SAS Federation Server can access data in several third-party relational databases, including DB2, Greenplum, ODBC data sources, Oracle, and Teradata.

The relational database drivers read, update, and create tables for those third-party relational databases on behalf of the Federation Server client. Each driver supports most of the FedSQL functionality. The Federation Server drivers support native database functionality by using the SQL dialect that is implemented by the third-party databases. For details about supported functionality and compatibility guidelines, see the specific data source reference:

- “DB2 Reference” on page 110
- “Greenplum Reference” on page 117
- “ODBC Reference” on page 125

Chapter 2

Upgrading SAS Federation Server

Overview

To upgrade to SAS Federation Server 3.2, you must perform certain tasks to update your databases and configuration files. There are two options for upgrading to SAS Federation Server 3.2:

- **Migration:** Migration is the process in which you upgrade your content and configuration from an earlier version of SAS Federation Server to run in SAS Federation Server 3.2. Perform a migration when you are upgrading to a different version platform (for example, from 2.1.x to 3.x). If you are using Federation Server 2.1.x, use the Migration tasks to update your databases to be compatible with SAS Federation Server 3.2.
- **Promotion:** Promotion is the movement of selected content from Federation Server 3.1.x to a current configured installation of SAS Federation Server on the same version platform (for example, from 3.1.x to 3.2). Use promotion if you are currently using Federation Server 3.1.x and are upgrading your databases to SAS Federation Server 3.2.

Migrating to SAS Federation Server 3.2

Overview

SAS Federation Server 3.2 installation contains a migration utility that enables you to migrate Federation Server 2.1.x. The Windows installation contains a shortcut labeled **Migrate 2.1.x Database**, and the UNIX installer uses the **migrate** option with **dfsadmin**. Each option requires that you point the migration utility to the 2.1.x installation directory.

Invoking the Migration Utility — Windows

The migration process involves copying data from your old installation into the current installation and upgrading it from the 2.x format to the 3.x format. When running the migration utility, you will be asked for the location of the 2.1.x installation as shown in the following task.

1. After installing SAS Federation Server, run the **Migrate 2.1.x Database** tool, which launches a mini-installer.
2. Click **Next** in the **Setup** dialog box.
3. At **Choose Old Install Location**, browse to the root directory for Federation Server 2.1.x (for example, *C:\Program Files (x86)\DataFlux\FederationServer\2.1*).
4. Click **Next** and **Start** to begin the migration.
5. When the migration completes, click **Next** and **Finish**.

Note: Migration results are written to the **`dfs_log_213_310.xml`**file located in the **`\etc\migrate`** directory.

Invoking the Migration Utility — UNIX

The UNIX utility, **dfsadmin**, contains a migration tool to migrate a Federation Server system database (syscat.tdb) from 2.1.x to 3.2. Use the following syntax to invoke migration:

```
./bin/dfsadmin migrate <2.1.x_installation_directory>
```

The migration tool reads from the 2.1.x Federation Server configuration file to determine where the 2.1.x system database (syscat.tdb) file is located and then migrates it to the location for release 3.1. Migration results are written to a log file, **`dfs_log_213_310.xml`**, located in **`/etc/migrate`**.

Promoting Content to SAS Federation Server 3.2

To promote SAS Federation Server databases:

1. Stop the 3.1 servers using **Windows Services Manager**, the **Start** menu, or the dfsadmin utility for UNIX.

2. Copy the 3.1 SYSCAT.TDB and SQL_LOG.TDB files to the 3.x installation location (for example, <*SAS Home*>*SASFederationServer**3.x**var*).
3. Start the 3.x servers using **Windows Services Manager**, the **Start**menu, or the dfsadminutility for UNIX.

Note: If any custom configuration changes were made to the 3.1 configuration files, you must update the new 3.x configuration files with these configurations. For more information, see SAS Federation Server Configuration Reference on page 31 .

ODBC Drivers

Overview

SAS Federation Server 3.2 is installed with release 7.1 of the ODBC drivers. The installation directory for Windows has changed to **`C:\Program Files(x86)\DataFlux\ODBC\7.1`**. All of the existing ODBC DSNs should use the new 7.1 ODBC drivers and modifications are not necessary. If you are running ODBC on UNIX, you must make some minor modifications.

Modifying ODBC for UNIX

To use the new 7.1 ODBC drivers, take one of the following actions:

- Configure the existing **odbc.ini** file to point to the new drivers in **`<3x_SASHome>/SASFederationServer/3.x/fedserver/lib/fx*27.so`**. For example, change the **Driver**=value, to the following:

  ```
  Driver=<3x_SASHome>/SASFederationServer/3.x/fedserver/lib/FXora27.so.
  ```

- Use the **dfdbconf** tool to configure the data sources to use the new drivers. You should already have ODBCINI and ODBCINST set from a previous installation. If using **dfdbconf**, you must export the ODBCINI and ODBCINST environment variables again. Here are examples:

  ```
  export ODBCINI=<3x_SASHome>/SASFederationServer/3.x/fedserver/etc/odbc.ini
  ```

  ```
  export ODBCINST=<3x_SASHome>/SASFederationServer/3.x/fedserver/etc/odbcinst.ini
  ```

  For more information, see Configuring ODBC Connections on page 23 .

Chapter 3

Configuring SAS Federation Server

Overview

This chapter focuses on post-installation configuration of SAS Federation Server for Windows and UNIX environments. Also included is a configuration reference that covers all of the possible configuration options for SAS Federation Server. SAS Federation Server Manager configuration is also covered in this chapter. SAS Federation Server Manager is the graphical interface for administration of SAS Federation Server.

Configure Temporary Storage for SAS Utility Files

The Directory Location

SAS Federation Server and other SAS applications create temporary utility files that are written to the default **/tmp** or **/temp** directory that is set in your environment. It is recommended that you specify a location for these files to ensure that there is enough space for processes, such as threaded applications, to create utility files. Use one of the procedures below to configure an environment variable to accommodate the utility directory and files. The name of the SAS utility directory is determined by the following:

```
SAS_util<serial><pid>_<node>
```

- <serial> is a unique 4-to 6-digit hexadecimal serial number that distinguishes each directory from the other directories that are created by the same process.
- <pid> is the process ID number, which is represented as an 8 digit hexadecimal number.
- <node> is the name of the host,or machine on which the process is running.

UNIX

In UNIX, set the location for utility files using TKUTILLOC in an export statement:

```
export TKUTILLOC=~directory_1/dfs
```

The utility directory and files are created in the specified directory. When SAS Federation Server is started, you should see a directory similar to **SAS_util000100000EF0_devmachine** that contains *.utl files

Windows

In Windows, use the Control Panel to set the TKUTILLOC environment variable:

1. From the Control Panel, select **System and Security** and then select **System**.
2. Select **Advanced system settings** to open the System Properties window.
3. Click on environment variables.
4. Under system variables click on **New** and set TKUTILLOC as the variable name with the path to the directory that will store utility files. Utility files contain a .utl extension. For example: c:\temp\fedserver.

5. Click **OK** and start the Federation Server.

When SAS Federation Server is started, you should see a directory similar to **`SAS_util000100000EF0_devmachine`** that contains *.utl files.

Note: If you set the directory to a location that does not exist, TKUTILLOC does not create the directory and reverts to the default temporary directory. For example, on Windows, **`C:\Users\user_1\AppData\Local\Temp`**.

Configuring the Windows Environment

Overview

This section outlines the necessary configuration procedures and server tasks that you must complete following installation of SAS Federation Server in a Windows environment.

Federation Server Directory Permissions

The recommended directory permissions for SAS Federation Server installed on a Windows® platform are listed in the following table:

Directories	Users	Default Permissions
`SASHome`	Installer, Administrator	Full Control
`SASHome\fedserver\server instance`	Process user	Read and Execute, List Folder Contents
`SASHome\fedserver\server instance\var`	Installer, Administrator	Full Control
	Process user	Read, Write, List Folder Contents
	The user who backs up SAS Federation Server; Backup Administrator	Read, List Folder Contents
TranPath as specified in the server configuration file	Installer, Administrator	Full Control
	Process user	Read, Write, List Folder Contents
	The user who backs up SAS Federation Server; Backup Administrator	Read, List Folder Contents

Note: All other users have no access.

Configuring ODBC Connections

This section describes how to configure ODBC connections for SAS Federation Server in a Windows environment.

To access a database through ODBC with SAS Federation Server, a driver for ODBC for the specific database must be used. SAS Federation Server comes with the ODBC drivers for several databases. The database and database connection must also be configured as an ODBC data source.

To access a database through a vendor supplied client, the client must be installed and configured according to the vendor documentation.

To add an ODBC data source, use the ODBC Data Source Administrator in Microsoft Windows. To set up a new ODBC connection:

1. Click **Start** ⇨ **Control Panel**.
2. Double-click **Administrative Tools** ⇨ **Data Sources (ODBC)**.

   Note: In Windows 7, the view of the Control Panel can vary. If you do not see Administrative Tools when you open the Control Panel, click System and Security to access Administrative Tools, Data Sources (ODBC).
3. Click **Add**.
4. In the ODBC Data Source Administrator window, select the driver that is appropriate for your data source.
5. In the ODBC Driver Setup dialog box, enter the Data Source Name, Description, and database-specific configuration options. These values are required, and can be obtained from your database administrator.

Use the vendor supplied client configuration utility for connections that are not ODBC.

Starting and Stopping the Windows Service

The SAS® Federation Server runs as a Windows service that is accessible through Administrative Tools - Services in the Control Panel or Management Console. To access the service:

1. Click **Start** ⇨ **Settings** ⇨ **Control Panel**
2. Double-click **Administrative Tools** ⇨ **Computer Management**.
3. Expand the **Services and Applications** folder.
4. Click **Services**.
5. Click**SAS Federation Server**.
6. Click either **Stop the service** or **Restart the service**.

Note: Version number differences between the database schema and the Federation Server itself can terminate the start process. Major version numbers must match. The minor version number of the schema can be 1 less than the minor version number of the server.

Modifying the Service Log On

At installation, SAS Federation Server service is configured to start using the local system account. Because this account can have some restrictions, such as accessing

network drives, it is suggested that you modify the service log on account to an account that has the appropriate privileges to run SAS Federation Server.

To modify the SAS Federation Server service log on:

1. Select **Control Panel** ⇨ **Administrative Tools**.
2. Double-click **Services**, and select the **SAS Federation Server service**.
3. Click the **Log On** tab, select **This account**, and enter Account and Password credentials for a user with administrative privileges.

Configuring the UNIX Environment

Overview

This chapter outlines the necessary configuration procedures and server tasks that you must complete following installation of SAS Federation Server in a UNIX environment.

UNIX File System and Directory Permissions

The recommended file permissions for Federation Server installed on a UNIX platform are listed in the following table:

Directories	Users	Default Permissions
SASHome	Installer, Administrator	Read, Write, Execute
SASHome/SAS Federation Server	Process user	Read, Execute
SASHome/SAS Federation Server//server instance/var	Installer, Administrator	Read, Write, Execute
	Process user	Read, Write, Execute
	The user who backs up SAS Federation Server; Backup Administrator	Read, Execute
TranPath as specified in the server configuration file (dfs_serv_common.xml)	Installer, Administrator	Read, Write, Execute
	Process user	Read, Write, Execute
	The user who backs up SAS Federation Server; Backup Administrator	Read, Execute

Note: All other users have no access.

Setting Environment Variables

Overview

Before configuring SAS Federation Server driver software, set environment variables as outlined in the following sections.

Set the LANG Environment Variable

If using BASE data sets with the Federation Server, the LANG environment variable must be set before bringing up the Federation Server. This environment variable is primarily needed for the VALIEDATEFMT table.

Most UNIX or Linux systems use the LANG variable to specify the desired locale and this variable is often already set in your environment. Locale names vary among different UNIX or Linux operating systems, so use a LANG value that is supported by your version of UNIX or Linux.

- Invoke the locale command to show your current locale.
- Use **`locale –a`** to display a list of all the locales that are currently installed on the machine.

For more information about setting locale environment variables, consult the documentation for your operating system.

Setting Environment Variables for DBMS-Based Drivers

Before configuring your Federation Server, you should determine the following information about your DBMS:

- The version or release of the DBMS client shared libraries installed on your operating system. This is important due to potential incompatibilities between DBMS versions or releases.
- The location of the DBMS client shared libraries. This is important so that the correct client libraries can be loaded.

Note: The steps outlined in this chapter assume that the ODBC drivers were installed during installation of SAS Federation Server.

Refer to the following sections for detailed instructions on configuring your environment to interface with your Federation Server driver software:

- “SAP Reference ” on page 137
- “DB2 Reference” on page 110
- “ODBC Reference” on page 125
- “Oracle Reference” on page 131
- “Teradata Reference” on page 154

SAS Federation Server Driver for SAP®

SAP® software requires extensive configuration before it can be used. For more information, see See “Installing and Configuring the SAS Federation Server Driver for SAP” on page 142 .

SAS Federation Server Driver for DB2®

The SAS Federation Server Driver for DB2 uses shared libraries, referred to in UNIX as shared objects. You must add the location of the shared libraries to one of the system

environment variables, and, if necessary, indicate the DB2 version that you have installed at your site. Before setting the environment variables as shown in the examples below, you must also set the following environment variables:

- The INSTHOME environment variable must be set to your DB2 home directory.
- The DB2DIR environment variable should also be set to the value of INSTHOME.
- The DB2INSTANCE environment variable should be set to the DB2 instance configured by the administrator.

AIX	
Bourne Shell	`$ LIBPATH=$INSTHOME/lib:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH $INSTHOME/lib:$LIBPATH`

HP-UX and HP-UX for the Itanium Processor Family Architecture	
Bourne Shell	`$ SHLIB_PATH=$INSTHOME/lib:$SHLIB_PATH` `$ export SHLIB_PATH`
C Shell	`$ setenv SHLIB_PATH` `$INSTHOME/lib:$SHLIB_PATH`

Linux for Intel Architecture, Linux for x64, Solaris, and Solaris for x64	

Bourne Shell	`$`
	`LD_LIBRARY_PATH=$INSTHOME/lib:$LD_LIBRARY_PATH`
	`$ export LD_LIBRARY_PATH`

C Shell	`$ setenv LD_LIBRARY_PATH` `$INSTHOME/lib:$LD_LIBRARY_PATH`

SAS Federation Server Driver for ODBC

To configure ODBC data sources, you might have to edit the .odbc.ini file in your home directory. Some ODBC Driver vendors allow system administrators to maintain a centralized copy by setting the environment variable **ODBCINI**. Please refer to your vendor documentation to find specific information. Additional information for ODBC can be found in "Configuring ODBC Connections" on page 16.

The Drivers for ODBC are ODBC API–compliant shared libraries, referred to in UNIX as shared objects. You must add the location of the shared libraries to one of the system environment variables so that drivers for ODBC are loaded dynamically at run time. You must also set the **ODBCHOME** environment variable to your ODBC home directory before setting the environment variables as shown in the following examples.

Linux for Intel Architecture and Linux for x64	
Bourne Shell	`$ LD_LIBRARY_PATH=$ODBCHOME/lib:$LD_LIBRARY_PATH` `$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH $ODBCHOME/lib:$LD_LIBRARY_PATH`

Solaris and Solaris for x64	
Bourne Shell	`$ LD_LIBRARY_PATH=$ODBCHOME/lib:$LD_LIBRARY_PATH` `$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH $ODBCHOME/lib:${LD_LIBRARY_PATH}`

AIX	
Bourne Shell	`$ LIBPATH=$ODBCHOME/lib:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH $ODBCHOME/lib:${LIBPATH}`

HP-UX and HP-UX for the Itanium Processor Family Architecture	
Bourne Shell	`$ SHLIB_PATH=$ODBCHOME/lib:$SHLIB_PATH` `$ export SHLIB_PATH`
C Shell	`$ setenv SHLIB_PATH $ODBCHOME/lib:${SHLIB_PATH}`

Setting ODBC for Greenplum

To use ODBC with Greenplum, you need to set the ODBCINST and ODBCINI environment variables to point to the configuration files that are installed with SAS Federation Server.

1. Set the ODBCINST environment variable to the **odbcinst.ini** file located in the Federation Server installation path:

   ```
   export ODBCINST=$installpath/fedserver/etc/odbcinst.ini
   ```

2. Set the ODBCINI environment variable to the **odbc.ini** file located in the Federation Server installation path:

   ```
   export ODBCINI=$installpath/fedserver/etc/odbc.ini
   ```

SAS Federation Server Driver for Oracle

Connecting to an Oracle Database

You can connect to any Oracle server from SAS Federation Server using the Oracle driver, client versions 10g or 11g. If you are using Oracle client version 11g, run these additional commands to successfully connect with Oracle from SAS Federation Server:

```
cd
/installation_directory/fedserver/lib
cp tkeora11.so tkeora.so
```

Note: For HP64, the file extensions are .sl.

Setting the ORACLE_HOME Variable

In order to use the SAS Federation Server Driver for Oracle, you must set the ORACLE_HOME environment variable. In addition, you must make sure that the shared library path variable (the name of this variable is operating system dependent) points to where the Oracle shared libraries are located. This is required since the SAS Federation Server Driver for Oracle executable uses Oracle shared libraries and needs to know where they are located at your site.

The following are examples for the various operating systems:

AIX	
Bourne Shell	`$ LIBPATH=$ORACLE_HOME/lib:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH=$ORACLE_HOME/lib:$LIBPATH`

HP-UX and HP-UX for the Itanium Processor Family Architecture	
Bourne Shell	`$ SHLIB_PATH=$ORACLE_HOME/lib:$SHLIB_PATH` `$ export SHLIB_PATH`
C Shell	`$ setenv SHLIB_PATH $ORACLE_HOME/lib:$SHLIB_PATH`

Linux for Intel Architecture, Linux for Itanium-based Systems, Solaris, and Solaris for x64	
Bourne Shell	`$ LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH` `$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH $ORACLE_HOME/lib:$LD_LIBRARY_PATH`

SAS Federation Server Driver for Teradata

Access to Shared Libraries

The SAS Federation Server Driver for Teradata uses shared libraries, referred to in UNIX as shared objects. You must add the location of the shared libraries to one of the system environment variables.

AIX	
Bourne Shell	`$ LIBPATH=TERADATA-CLIENT-LOCATION:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH TERADATA-CLIENT-LOCATION:$LIBPATH`

HP-UX	
Bourne Shell	`$ SHLIB_PATH=TERADATA-CLIENT-LOCATION:$SHLIB_PATH` `$ export SHLIB_PATH`
C Shell	`$ setenv SHLIB_PATH TERADATA-CLIENT-LOCATION:$SHLIB_PATH`

HP-UX for the Itanium Processor Family	

Bourne Shell	`$ SHLIB_PATH=TERADATA-CLIENT-LOCATION:$SHLIB_PATH`
	`$ export SHLIB_PATH`
	`$ LD_PRELOAD=/usr/lib/hpux64/libpthread.so.1`
	`$ export LD_PRELOAD`

C Shell	`$ setenv SHLIB_PATH TERADATA-CLIENT-LOCATION:$SHLIB_PATH` `$ setenv LD_PRELOAD /usr/lib/hpux64/libpthread.so.1`

Linux for Intel Architecture, Linux for x64, Solaris, and Solaris for x64	
Bourne Shell	`$ LD_LIBRARY_PATH=TERADATA-CLIENT-LOCATION:$LD_LIBRARY_PATH` `$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH TERADATA-CLIENT-LOCATION:$LD_LIBRARY_PATH`

TTU 8.2 and HP-UX

HP–UX users with TTU 8.2 must create two symbolic links from the /usr/lib/pa20_64 directory with the following commands:

```
$ ln -s /usr/lib/pa20_64/libicudatatd.sl libicudatatd.sl.34
$ ln -s /usr/lib/pa20_64/libicuuctd.sl libicuuctd.sl.34
```

Configuring ODBC Connections

Overview

To access a database through ODBC with SAS Federation Server, an ODBC driver for the specific database must be used. SAS Federation Server is delivered with branded ODBC drivers for several databases. You can opt to use one of these ODBC drivers or you can use a third party driver. The database must also be configured as an ODBC data source when using an ODBC driver.

Configuring ODBC Connections Using the SAS Federation Server Drivers

SAS Federation Server includes an ODBC configuration tool used to configure the ODBC data sources that are installed with SAS Federation Server.

Note: If you are upgrading SAS Federation Server, see Modifying ODBC for UNIX on page 11 in *Upgrading SAS Federation Server*.

ODBC Configuration Tool

Use the ODBC configuration tool, **dfdbconf**, to add new data sources to your ODBC configuration.

To add a new data source:

1. From the root directory of SAS Federation Server installation, run: **`./bin/dfdbconf`**.
2. Select **`A`** to add a data source.
3. Select a template for the new data source by choosing a number from the list of available drivers.
4. Set parameters for the driver as you are prompted to do so. The new data source is added to the **`odbc.ini`** file.

TIP You can also use **`dfdbconf`** to modify an existing data source if it is no longer needed.

Testing the Connection

Once you have added all of your data sources, use the interactive ODBC Viewer application, **`dfdbview`**, to test your connection. For example, if you added a data source calledmy_oracle, run **`./bin/dfdbview my_oracle`** (from the installation root) to test the connection. You might be prompted for a user name and password. If the connection succeeds, you will see a prompt from which you can enter SQL commands and query the database. If the connection fails, **`dfdbview`** displays error messages describing one or more reasons for the failure.

Configuring ODBC Connections Using Third Party ODBC Drivers

- To access a database through ODBC with SAS® Federation Server, an ODBC driver for the specific database must be used. The database must also be configured as an ODBC data source when using an ODBC driver.
- To access a database through a vendor supplied client, the client must be installed and configured according to the vendor documentation.

- Verify the connection with a third-party client tool prior to attempting connection through SAS Federation Server.

unixODBC Driver Manager

unixODBC is an open source product that implements the ODBC API. If unixODBC is required and not already installed, visit **http://www.unixODBC.org** and download the required software.

The following configurations are required when using the unixODBC driver manager with SAS Federation Server:

1. Include unixODBC in the **PATH** and **LD_LIBRARY_PATH**.
2. The **odbcini** and **odbcinst.ini** installed with SAS Federation Server are for use with the ODBC driver manager, also installed with SAS Federation Server. Since the third-party ODBC driver will likely use the unixODBC driver manager, you will need to update the **odbcini** and **odbcinst** files used by unixODBC and update the OBCDINI environment variable accordingly.
3. **DM_UNICODE=utf-16** is required in the advanced options of any data service that is used with the driver manager.

Use the vendor supplied client configuration utility for non-ODBC connections. For more information about configuring third-party databases, See "Setting Environment Variables for DBMS-Based Drivers" on page 18 .

Starting and Stopping the Server Daemon

Start and stop the SAS Federation Server daemon using the **dfsadmin** application, located in the **bin** directory of the installation root. This application is run using the command syntax: **./bin/dfsadmin yourcommand** where **yourcommand** is one of the following:

To start SAS Federation Server: **./bin/dfsadmin start**

To stop SAS Federation Server: **./bin/dfsadmin stop**

Additional information for the **dfsadmin** application can be found in the "dfsadmin Utility" on page 63 topic.

Configuring Server Accounts

Overview

SAS Federation Server uses the following accounts for administration, authentication and data authorization:

- "The System User Account" on page 25

  The system user account is the most privileged account for SAS Federation Server.

- "Administrator Account" on page 26

  An administrator account is a user account created in Authentication Server, and then granted ADMINISTER privilege on SAS Federation Server.

- "The Trusted User Account" on page 26

The trusted user account establishes a trust relationship between the SAS Federation Server and the Authentication Server.

- "Shared Logins" on page 27

  Shared logins help manage user connections from SAS Federation Server to the data. With shared logins, users can access data without needing individual logins to access the data source.

The System User Account

Introduction

The SYSTEM user is a privileged account which means that it carries more privileges than an administrator account. There is nothing on SAS Federation Server that the system account cannot do because the account has implicit privileges to all user and data objects. The operating system process that invokes and runs SAS Federation Server is automatically a system user. By default, the account used to install SAS Federation Server is registered as a system user account. Other user accounts can also be registered as system users in the **dfs_serv_common.xml** configuration file.

The system user should identify users who will be administrators of SAS Federation Server, and make them administrators by granting them the ADMINISTER privilege on the server object. Like SYSTEM users, administrators are unconditionally and implicitly granted all privileges on SAS Federation Server. However, if these users are revoked their ADMINISTER privilege, then they become standard users that have privileges granted to and denied from them. A SYSTEM user can never be denied privileges.

If a Data Source Name (DSN) is created by either the system user or an administrator, the DSN is created using the AS ADMINISTRATOR clause, which means that the ADMINISTRATOR role owns the DSN, not the individual creating it. Therefore, if the administrator user is later removed from the system, the DSN will not be deleted with the user.

Use the system user account to define one or more administrators for SAS Federation Server. As a best practice, all configuration and administration should be performed by the administrator.

Configuring the SystemUsers OptionSet

The SystemUsers option set defines the system user accounts for the SAS Federation Server. When system users grant or deny privileges to others, the grantor is reflected in the system tables as the system user ID. Each system user should be a domain-qualified user name.

```
<SystemUsers>
   <Option name="Account">domain\uid1</Option>
   <Option name="Account">domain\uid2</Option>
</SystemUsers>
```

Altering the System User Account

You can add or remove system users by updating configurations in the SystemUsers option set explained above.

Administrator Account

Overview

An administrator account is a user account created in Authentication Server, and then granted ADMINISTER privilege on the SAS Federation Server. Only system users can grant the ADMINISTER privilege.

Administrators have implicit privileges to perform every other action including the following:

- create and drop data source names (DSN)
- grant and deny privileges to other accounts
- create and drop data services, catalogs, and schemas

Setting the ADMINISTER Privilege

To define a user as an administrator for SAS Federation Server, grant the ADMINISTER privilege to their account using the following syntax:

```
GRANT serverpriv ON servername TO "user-ID"
GRANT administer ON FedServer1 TO "user1"
```

For further details, reference the GRANT and DENY DDL statements on page 183 .

The Trusted User Account

Overview

The trusted user account is set up in the Authentication Server and configured in SAS Federation Server using the ALTER SERVER statement on page 159 or SAS Federation Server Manager. Use this account to establish a trust relationship between the SAS Federation Server and the Authentication Server. The trust relationship is required for certain features to work such as definer's rights views.

A few notes about the Trusted User account:

- A trusted user is a user ID that has to be able to authenticate using the authentication method that is deployed for the installation.
- As stated above, this account is used only by SAS Federation Server to connect to Authentication Server in certain scenarios like definers rights views. You would never log on using this account.
- The trusted user should not be a system or administrative user on Authentication Server or SAS Federation Server.

Using DDL to Configure the Trusted User

Use the ALTER SERVER statement on page 159 to define the trusted user.

```
ALTER SERVER {OPTIONS(xset TRUSTED_USER_UID 'user-ID',
xset TRUSTED_USER_PWD 'password')}
```

Note: User IDs and passwords must be added or dropped in pairs.

Configure the Trusted User in Federation Server Manager

You can also configure a trusted user account using SAS Federation Server Manager.

To define the trusted user:

1. Locate the SAS Federation Server object in the tree and select the **Action Menu** in the upper left corner.
2. Using the action menu, select **Properties** to display the server properties dialog box.
3. Click the **Security** tab in the Federation Server Properties dialog box.
4. Set the Trusted user ID and password and click **OK**.

Shared Logins

Overview

Using Shared Logins, users can access data when they do not or should not need to know the credentials used to access this data. Shared Logins help manage user connections from SAS Federation Server and outside of the database. Only one set of credentials is used to access data for groups of people who will never know the credentials that are being used. Authorization management occurs and is managed at a layer above the database. Using shared logins is optional. If using shared logins to connect to data sources, they must be configured in SAS Federation Server before they will function properly in a DSN.

Shared Login Manager

When using shared logins to authenticate users to a data source, users do not need to know the credentials that they are using because the shared login manager is retrieving credentials for the user that is logged on and providing the credentials to SAS Federation Server so that the server can connect the user to the database through the appropriate data service or DSN.

- The shared login account is created in Authentication Server which includes the login to be shared and its domain, consumers of the login, and a login key.
- The shared login manager and login key is configured in SAS Federation Server using administrative DDL or SAS Federation Server Manager.

  The shared login manager login identifies the credentials used to connect to the Authentication Server and retrieve the shared login. It is required that the shared login manager be a manager of that shared login in Authentication Server. The shared login is not directly readable by consumers of the shared login. It can be read only by managers or the owner. The shared login manager's password is encrypted.

  The shared login key identifies which shared logins created in Authentication Server are available to a SAS Federation Server instance. The key defined in SAS Federation Server must match the key that is part of the shared login definition in the Authentication Server.
- After the shared login manager account is configured, the account information can be referenced from DSNs in SAS Federation Server. The DSN is configured with a credentials search order of SHARED.

Configure shared logins in SAS Federation Server using the ALTER SERVER DDL statement. You can also configure a shared login though SAS Federation Server Manager, using the Security dialog box located in the properties of a Federation Server object.

Shared Login Manager Configuration using DDL

Use the ALTER SERVER statement on page 159 to set shared logins and to configure the shared login key. For example,

```
alter server {options xset sharedloginkey 'DefaultKey'}
alter server {OPTIONS(xset SHAREDLOGINMANAGER 'manager user id',
xset SHAREDLOGINPASSWORD 'manager password')}
```

Note: User IDs and passwords must be added or dropped in pairs.

Shared Login Configuration using Federation Server Manager

1. Locate the Federation Server object in the tree and select the **Action Menu** in the upper left corner.
2. From the action menu, select **Properties** to display the server properties dialog box.
3. Select the **Security** tab and configure the shared login with the following information:
   - Shared login manager user ID
   - Shared login manager password
   - Shared login key

After shared logins are configured, they can be referenced to establish connections to data sources. Reference the "Configuring DSNs" on page 89 topic for additional information.

Best Practices for Setting Shared Logins

Overview

Shared logins are created in the Authentication Server when there is a need to share logins among multiple users. Logins must be owned by users, not groups. Therefore, shared logins are the only mechanism to provide this functionality.

Shared logins consist mainly of the login and domain to share, and the consumers who use that login. The consumers will typically list one or more groups, which is a best practice. A conflict can arise when a particular user is in a consumer group (directly or indirectly) of multiple principal maps for the same domain. The following scenarios explain how shared login conflicts are resolved.

Scenario 1: Application Users

In the following scenario, an application exists which requires the use of a particular set of database credentials to access its protected data. For example, an HR application has data content stored in Oracle and DB2.

To manage credentials:

1. Identify all the users of the HR application. The users might have different roles or data access privileges, but they all need to access the data. These users (or subgroups) will all be placed in the group HR_USERS.
2. Create a shared login for each domain. In this case, the administrator would create an HR_ORACLE and HR_DB2 shared login. For both shared logins, the administrator would specify the HR_USERS group as a consumer member of the shared login. Each shared login would contain the appropriate principal and domain for the database.

3. Specify the GROUP option to qualify the users with the shared login, either in the DSN itself, or on the connection string specifying the DSN. In this case, the GROUP option would be HR_USERS.
4. Set authorizations on different users and groups to control which set of users can perform which operations, for example SELECT versus UPDATE versus DELETE. All of the users and groups should be members of the HR_USERS group.

At connect time, the HR_USERS group is used to identify the correct shared login for each underlying database connection. If the connecting user were a consuming member of another shared login, the GROUP value would properly identify which one to use.

Algorithm When Using the GROUP Option

Shared logins are initially considered candidates for outbound credentials selection if the domain and shared login key match. If the domain is empty, shared logins for any domain initially qualify. This also applies to the shared login key, which is configured as part of SAS Federation Server.

If the GROUP connection string option is specified (which is derived from the consumer group in the DSN configuration), then only maps where the group is a direct or indirect consumer will be considered a candidate for outbound credentials selection. The basic algorithm selects a map based on the closeness of the specified group to the map.

Candidate Map Processing

1. If the user is not a direct or indirect member of the shared login consumer group, the map is not a candidate; otherwise,
2. If the GROUP is not a direct or indirect consumer of the map, the map is not a candidate; otherwise,
3. The distance from the GROUP to the map is computed by following the group member-of relationship up to the group that is the direct consumer. The candidate map is retained if and only if the distance is less than or equal to the current minimum distance to the map. The current minimum distance is updated.

After all candidate maps have been processed,

1. if exactly one has been retained, return OK and the associated credentials; or
2. if two or more were retained, check the closest two, and return the credentials associated with the closer of the two or return an error if the distances are the same unresolved conflict; or
3. if no maps have been retained, then return OK but empty credentials.

Scenario 2: Organized Consuming Users

In the following scenario, the administrator has organized the users based on company organization or another classification.

The administrator wants to use this relationship to have users qualify for a particular shared login, for example:

- The administrator wants to grant access to Oracle account EXECUTIVE_USER to his most privileged users, identified by the MARKETING_EXECUTIVE group.
- The administrator wants to grant access to Oracle account MARKETING_USER to members of the marketing division in the company, identified by the MARKETING group.

- The administrator wants to grant access to Oracle account STANDARD_USER to all other known users in the system, identified by the USERS group.
- The administrator has created groups which reflect the company's organizational chart. The MARKETING group reflects all members of the marketing organization, with the MARKETING_EXECUTIVE group included as a member of the MARKETING group.

The administrator creates shared logins for the EXECUTIVE_USER, MARKETING_USER and STANDARD_USER Oracle accounts. Next, he assigns MARKETING_EXECUTIVE, MARKETING and USERS consuming groups, respectively, to these shared logins.

Then, the shared login chosen would be as follows:

- For members of the MARKETING_EXECUTIVE group, they would be closest to the shared login identified by that group, even though they were likewise members of the MARKETING and USERS groups. Therefore, this set of users would consume the EXECUTIVE_USER Oracle account.
- For members of the MARKETING group, they would be closest to the Shared Login identified by that group, even though they were likewise members of the USERS group. Therefore, this set of users would consume the MARKETING_USER Oracle account.
- All other known users would qualify only for the shared login identified by the USERS group. Therefore, this set of users would consume the STANDARD_USER Oracle account.

In this scenario, the administrator would not use the GROUP option, since the option accepts only a single value, and no single value works for all users. The administrator would omit the GROUP option and allow a closeness algorithm to identify which shared login to use.

Algorithm When No GROUP Option is Specified

If the GROUP connection string option is not specified, then all maps where the user is a direct or indirect consumer will be considered a candidate for outbound credentials selection. The basic algorithm selects a map based on the closeness of the specified user to the map, or USERS if the user is not a consumer, and finally PUBLIC if USERS is not either. If the user is PUBLIC, selection is done based on PUBLIC alone.

Candidate Map Processing For a User

1. If the user is not a direct or indirect consumer of the map, the map is not a candidate; otherwise,
2. The distance from the user to the map is computed by following the group member-of relationship up to the group that is the direct consumer. The candidate map is retained if and only if the distance is less than or equal to the current minimum distance to the map. The current minimum distance is updated.

After all candidate maps have been processed,

1. if exactly one has been retained, return the associated credentials; or
2. if two or more were retained, check the closest two, and return the credentials associated with the closer of the two or return an error if the distances are the same (unresolved conflict); or
3. Continue on to Candidate Map Processing for USERS. If the user is PUBLIC, go to Candidate Map Processing for PUBLIC.

Candidate Map Processing For USERS

- If USERS is a direct consumer of the map, then the candidate map is retained.
- If a candidate map has been retained already, return an error (unresolved conflict).
- After all candidate maps have been processed, if exactly one has been retained, return OK and the associated credentials; or
- Continue on to Candidate Map Processing for PUBLIC.

Candidate Map Processing For PUBLIC

- If USERS is a direct consumer of the map, then the candidate map is retained.
- If a candidate map has been retained already, return an error (unresolved conflict).
- After all candidate maps have been processed, if exactly one has been retained, return OK and the associated credentials; or
- Return OK but empty credentials if no candidate maps retained.

Path Length Computation Details

If USERS or PUBLIC is a member of another group, and that group is a map consumer, direct or indirect, the path length is not incremented when traversing from the user to the map. For the purposes of map selection, this effectively makes placing either of these two groups in another group a shorthand way to place all users in that group.

SAS Federation Server Configuration Reference

Locale Support

At this time, SAS Federation Server supports the English, United States of America (en_US) locale. The following table outlines the character representations for output (display) format. There are no deviations from these formats:

Character Type	Format
Number	ddddd.ffffffff
Date	yyyy-mm-dd
Time	hh:mm:ss
Timestamp	yyyy-mm-dd hh:mm:ss[.ffffffff]

Note: Configure database drivers and clients to match this behavior to ensure that conversions are handled correctly.

Key Configuration Files

The following table lists the key configuration files for SAS Federation Server:

File or Script Name	Description
dfs_serv.xml, **dfs_serv_common.xml**	These are the core configuration files for SAS Federation Server. They specify the system users, the location of the internal database, and other key configuration settings necessary for proper functionality of SAS Federation Server. These configuration files are located in the **/etc** directory of the Federation Server installation path. Detailed configuration information is presented in the Configuration Options on page 34.
dfs_entities.dtd	The **dfs_entities.dtd** file contains the values that were supplied during the install process. These values are referenced by other configuration files such as **dfs_serv.xml** and **dfs_serv_common.xml** files.
dfs_log.xml	This is the logging facility configuration file for SAS Federation Server. It specifies logging options for SAS Federation Server from information-only to debug and trace. This file is installed in the **/etc** directory of the installation path.For more information, see Server Logging Configuration on page 59..
dfs_log_SQL_Logging.xml	This is the configuration file that is used to facilitate SQL Logging. This file is located in the **/etc** directory of the installation path. For additional information, see"SQL Logging" on page 49 .
dfs_pwencode.xml	This is the logging file used when running the tool to encrypt a password. The logs are located in the **/var** directory of the Federation Server installation path. See Utilities for SAS Federation Server on page 63 for additional information about password encryption.

About the Server Configuration Files

SAS Federation Server uses the **dfs_entities.dtd** file to store values that are supplied during installation. These values are used by the other configuration files, **dfs_serv_common.xml**, **dfs_serv.xml**, **dfs_log4sas.xml**, and **dfs_log_sql_logging.xml**. For example, a port number supplied during installation is recorded in **dfs_entities.dtd** and the port number option in **dfs_serv.xml** points to the definition in the .**dtd** file:

dfs_serv.xml	`<Option name="Port">&cfg.port;</Option>`
dfs_entities.dtd	`<!ENTITY cfg.Port "2171">`

About Option Names and Option Sets

Overview

The **dfs_serv_common.xml** and **dfs_serv.xml** configuration files consist of a combination of option names and option sets that are explained below.

Option Names

Option names specify a **name=value** pair as configuration file options. They can stand alone in the configuration file or are contained within an option set. Here are the different types of option name configurations that appear in **dfs_serv.xml**:

This specifies a simple name=value pair as a configuration option:

```
<Option name="XXX">yyy</Option>
```

Example: `<Option name="Port">21030</Option>`

This represents a name=value pair where the value is a space-delimited string of values:

```
<Option name="XXX">yyy zzz aaa</Option>
```

Example: `<Option name="AdminLoginManagementPolicy">ADD REMOVE UPDATE</Option>`

Option Sets

An option set consists of various options (option names). Option names that belong in an OptionSet will not be assessed correctly if they are placed outside of the OptionSet. For example, the SystemUsers option set requires at least one option name, Account, be defined in order to function properly:

```
<SystemUsers>
    <Option name="Account">domain\uid1</Option>
    <Option name="Account">domain\uid2</Option>
</SystemUsers>
```

Child Option Sets

Here is an example of an option set within an option set which is called a child option set. The License option set shown in the following example contains two options to define: Provider and Location (where the license provider and the location of the license file is specified):

```
<OptionSet name="License">
   <OptionSet
name="Primary">
      <Option name="Provider">SAS</Option>
      <Option name="Location">path_to_license_file</Option>
   </OptionSet>
</OptionSet>
```

For a complete list of configuration options for SAS Federation Server, refer to "Configuration Options" on page 34 .

Required Configuration Options

Overview

The following configurations are required for SAS Federation Server operations.

Security Provider

The SecurityProvider option set provides information about the Federation Server security provider, including the threaded kernel extension name and other information specific to the security provider. Configuration of the extension and database options are required.

```
<OptionSet name="SecurityProvider">
   <Option name="extension">extension_name</Option>
   <Option name="Database">database name</Option>
</OptionSet>
```

Authentication Server

The AuthenticationServer option set defines the location of the Authentication Server that serves as the back-end for SAS Federation Server. The Authentication Server is required to authenticate SAS Federation Server users and store user information such as logins and group memberships.

The option, IOM_URI contains connection information for the Authentication Server and uses the following format:

```
iom://<machine>:<port>;Bridge;CLSID=2D1BCDBF-F900-4CA9-85F6-95ECDBAF2122
```

```
<OptionSet name="AuthenticationServer">
   <OptionSet name="PrimaryServer">
       <Option name="URI">IOM-URI</Option>
   <OptionSet>
<OptionSet>
```

License

The License Option Set provides information about the type of licensing issued for SAS Federation Server. Within the Primary option set, specify the license provider and the path to the license file.

```
OptionSet name="License">
   <OptionSet name="Primary">
      <Option name="Provider">SAS</Option>
      <Option name="Location">C:\Federation Server\server1\etc\license</Option>
   </OptionSet
```

The Primary option set is a required configuration for operation of SAS Federation Server. You cannot start the server without a valid license.

Configuration Options

Overview

The following sections reflect the options that are available in the system configuration files, **dfs_serv.xml** and **dfs_serv_common.xml**. A **dfs_entities.dtd** may exist in specific configurations. The **dfs_entities.dtd** file contains the values

supplied during installation of SAS Federation Server. These values are referenced by the system configuration files.

AppendEnv OptionSet

Syntax

```
<OptionSet name="AppendEnv">
   <Option name="FIREBIRD">drive:\install_loc\firebird</Option>
</OptionSet>
```

Description

The AppendEnv option set locates the specified OS environment variable and appends the specified option to the environment variable's current value. If the environment variable does not exist, it is created and set to the specified value. The AppendEnv option does not add a delimiter between the existing and appended environment variable values. Therefore, if a delimiter is needed, it should be included at the beginning of the specified value.

TIP Each AppendEnv, PrependEnv and SetEnv option is processed entirely in the order in which they appear in the configuration file.

SetEnv OptionSet

Syntax

```
<OptionSet name="SetEnv">
   <Option name="FIREBIRD">[drive]:\install_dir\lib\fbembed</Option>
   <Option name="FIREBIRD_LOG">[drive]:\install_dir\var\log</Option>
   <Option name="FIREBIRD_TMP">[drive]:\FDS_Tmp</Option>
</OptionSet>
```

Description

This option set appears in dfs_serv_common.xml. The SetEnv option sets the OS environment variables to specific values. If the environment variable does not exist, it will be created and set to the option value. If the environment variable does exist, the value will be updated to the option value.

Set FIREBIRD_TMP as an environment option should the default database directory run out of space. Once the default directory has no available space, the engine switches to the directory specified in FIREBIRD_TMP.

TIP Each AppendEnv, PrependEnv and SetEnv option is processed entirely in the order in which they appear in the configuration file.

Memory Size Option

Syntax

```
<Option name="MemSize">nnnnn [(K|k|M|m|T|t)[(B|b)]]</Option>,
```

Example

```
<Option name="MemSize">1G</Option>
```

Description

The MemSize system option specifies the total amount of memory available for each SAS Federation Server session. If a setting is not specified, all system memory is available for use by SAS Federation Server. However, SAS Federation Server will use only as much memory as it needs to complete a process. Setting a value that is too low will result in out-of-memory conditions.

Transactional Data Store Options

Syntax

```
<Option name="FIREBIRD">drive:\install_dir\lib\fbembed</Option>
<Option name="FIREBIRD_LOG">drive:\install_dir\var\log</Option>
```

Description

The FIREBIRD environment variable specifies the location of the Transactional Data Store installation files.

The FIREBIRD_LOG environment variable specifies the location of the log files for Transactional Data Store. The configuration file generated during installation sets the FIREBIRD_LOG option to the var\log directory of the installation path. If FIREBIRD_LOG is not set, the federation server will default to one of two locations:

- **TranPath:** If the TranPath environment variable is set, FIREBIRD_LOG is set to the TranPath value.
- **ContentRoot:** If TranPath is not set, FIREBIRD_LOG is set to the ContentRoot value as defined in the configuration file.

TranPath Option

Syntax

```
<Option name="TranPath">directory</Option>
```

Description

The TranPath option identifies the location in which to store the Federation Server Database files.

By default, the Federation Server Database files are stored in ContentRoot as defined in the configuration file. Federation Server Database system files cannot be stored on remote file systems. If ContentRoot is a network file system or share, use the TranPath option to redirect the Federation Server Database files to a local directory on the machine where SAS Federation Server is installed.

Rules and Dependencies

Here are the rules and dependencies for the TranPath option:

- A relative directory specified in the ContentRoot tag is resolved against the server's working directory.
- A relative directory specified in the TranPath tag is resolved against the directory specified in the ContentRoot tag, or the working directory of the server if none is specified.
- If the TranPath option is omitted, it defaults to the directory specified in the ContentRoot tag, or the working directory of the server if none is specified.
- The TranPath directory is used to resolve the server's system catalog database name specified in the provider-specific SecurityProvider or Database tag for the Transactional Data Store security provider (**tkescfb**) provider.

PrependEnv OptionSet

Syntax

```
<OptionSet name="PrependEnv">
   <Option name="FIREBIRD">drive:\install_loc\firebird</Option>
</OptionSet>
```

Description

This option appears in dfs_serv_common.xml. The PrependEnv option will find the indicated OS environment variable and prepend the option value to the OS environment variable value. If the environment variable does not exist, it will be

created and set to the option value. The PrependEnv option will not add a delimiter of any sort between the existing and new environment variable value. If a semicolon (;) is needed, then the option value should include it at the end.

TIP Each AppendEnv, PrependEnv and SetEnv option is processed entirely in the order in which they appear in the configuration file.

SystemUsers OptionSet

Syntax

```
<SystemUsers>
   <Option name="Account">domain\uid1</Option>
   <Option name="Account">domain\uid2</Option>
</SystemUsers>
```

Description

This option appears in dfs_serv_common.xml and defines the system user account(s) that are given all privileges to SAS Federation Server including all user and data objects. This privilege cannot be revoked or denied. When system users grant or deny privileges to others, the grantor is reflected in the system tables as the SYSTEM user ID. Each system user should be a domain-qualified user name.

SecurityProvider OptionSet

Syntax

```
<OptionSet name="SecurityProvider">
   <Option name="extension">extension_name</Option>
   <Option name="Database">database path</Option>
</OptionSet>
```

Description

This is a required configuration for SAS Federation Server. The security provider option set provides information about the server's security provider, including the threaded kernel extension name and other information specific to the security provider.

FunctionDispatchManager Option

Syntax

```
<Option name="FunctionDispatchManager">tktsfd</Option>
```

Description

This option appears in dfs_serv_common.xml. The Function Dispatch Manager tells FedSQL to load an extension that implements SQL functions, including row-level security. This option should always be set to **tktsfd**.

Database Option

Syntax

```
<Option name="Database">syscat.tdb</Option>
```

Description

Identifies the name of the database (syscat.tdb) to be used for the Transactional Data Store security provider (**tkescfb**).

The database name is appended to the TranPath value as defined elsewhere in the configuration file. The default TranPath value for Windows is ***Drive*:\Program Files\SASHome\SasFederationServer*version*\var**, so by default, syscat.tdb will reside in that location. The default TranPath value for UNIX is **<install_dir>/SASHome/SASFederationServer/*version*/var**, so by default, syscat.tdb will reside in that location.

CAUTION:

The syscat database must reside on a local file system as indicated by the default installation paths above. If the **`var`** directory is pointed to a remote file system such as a network file share or storage area network (SAN), the syscat database cannot be created, resulting in multiple application errors.

ContentRoot Option

Syntax

```
<Option name="ContentRoot">content_root_path</Option>
```

Description

This option, configured in dfs_serv_common.xml, defines the content root for SAS Federation Server. The content root is used to resolve all relative pathnames specified in SAS Federation Server configuration, such as a schema path. It is recommended that the value for ContentRoot be set to an absolute, fully qualified path.

Rules

If the ContentRoot option is not set, files will be written to the install directory.

- Content root is absolute or relative to the install directory.
- TRACEFILEPATH is absolute or relative to content root.
- TRACEFILE names are resolved against the TRACEFILEPATH path. Paths that don't match are rejected. TRACEFILE is a parameter of the "Connection Options for DB2" on page 110 connection string.
- PRIMARYPATH paths in schema configuration options are absolute or relative to content root.
- SCHEMA=(PRIMARYPATH) connection string options are resolved against PRIMARYPATH schema configuration path.

Port Option

Syntax

```
<Option name="Port">port_number</Option>
```

Description

Indicates the port on which SAS® Federation Server will start.

Authentication Server OptionSet

Syntax

```
<OptionSet name="AuthenticationServer">
   <OptionSet name="PrimaryServer">
       <Option name="URI">IOM-URI</Option>
   <OptionSet>
<OptionSet>
```

Description

A required configuration for SAS® Federation Server, this option defines the location of the Authentication Server that serves as the back end for the .

IOM_URI contains connection information for the Authentication Server and uses the following format:

```
iom://<machine>:<port>;Bridge;CLSID=2D1BCDBF-F900-4CA9-85F6-95ECDBAF2122
```

For example, IOM_URI can be used to connect to an Authentication Server that is running on port number 21030 and machine name myhost. The CLSID for an

Authentication Server should always be 2D1BCDBF-F900-4CA9-85F6-95ECDBAF2122:

```
iom://yourserver.yourdomain.com:21030;Bridge;CLSID=2D1BCDBF-F900-4CA9-85F6-95ECDBAF2122
```

License OptionSet

Syntax

```
<OptionSet name="License">
   <OptionSet name="Primary">
      <Option name="Provider">SAS</Option>
      <Option name="Location">path_to_license_file</Option>
   </OptionSet>
```

Description

This option set, configured in dfs_serv_common.xml, provides information about the type of licensing issued for the . The Primary option set is a required configuration. Within the Primary option set, specify the license provider and the path to the license file. The location option points to a setinit file.

ObjectServerParms Option

Syntax

```
<Option name="ObjectServerParms">object_server_parameters</Option>
```

Description

Identifies IOM object server parameters. Values for object_server_parameters include:

- **Clientencryptionlevel = (EVERYTHING | CREDENTIALS | NONE)**— Specifies the client encryption level to use. Valid values include:
  - **NONE**—Nothing is encrypted.
  - **CREDENTIALS**— Login credentials are encrypted. Note: these are the login credentials used to authenticate to the SAS® Federation Server, and NOT logins used as outbound credentials to connect to third-party databases. It also does NOT include credentials passed in administration DDL, such as CREATE ACCOUNT REGISTRATION or CREATE USER.
  - **EVERYTHING**— All client/server communications are encrypted. Setting a value of EVERYTHING can affect server performance.

NetworkEncryptAlgorithm Option

Syntax

```
<Option name="NetworkEncryptAlgorithm">algorithm |
("algorithm1", "algorithm2", ...)</Option>
```

Description

algorithm | ("algorithm1", "algorithm2", ...) — Specifies the algorithm or algorithms that can be used for encrypting data that is transferred between a client and a server across a network. When you specify two or more encryption algorithms, use a space or a comma to separate them, and enclose the algorithms in parentheses. If more than one algorithm is specified, the client session negotiates the first specified algorithm with the server session. If the client session does not support that algorithm, the second algorithm is negotiated, and so on.

This is set with the CLIENTENCRYPTIONLEVEL OBJECTSERVERPARMS option. Possible values include: SASProprietary which is the default, and AES if implementing DFSecure.

For more information about Advanced Encryption Standard (AES), see the *Authentication Server Administrator's Guide*.

SelectStarExpansion Option

Syntax

```
<Option name="SelectStarExpansion">ALL</Option>
```

Description

This option modifies the behavior of the SELECT * expansion for table columns. The configuration options are ALL or VISIBLE. If set to ALL, the SAS® Federation Server attempts to expand SELECT * to all of the physical columns in the table and fails if the user does not have the SELECT privilege to one or more columns. If set at VISIBLE, which is the default value, SAS® Federation Server traverses the visible path, expanding the SELECT * privilege to those columns for which the user has the SELECT privilege.

Configuring SAS Federation Server Manager

Start the Configuration

SAS Federation Server Manager starts a configuration dialog box when you first login after installation. Use the following web address to start Federation Server Manager configuration again: **`http://host:port/fsmanager/web/config`**

For example: **`http://FedSvrManager:21077/fsmanager/web/config`**

CAUTION:

Before configuring SAS Federation Server Manager, the Federation Server service or daemon must be started.

1. After launching the configuration URL, you are redirected to a login page. Log in to SAS Federation Server Manager using the administrator user and password that was set during installation.

   TIP If you do not remember what the admin credentials are, look in **`fsmanager.cfg`** located at **Drive:\SASHome\FSManager*version**etc*** ⇨ . The password is case-sensitive.

2. After logging in, you are presented with an Initial Setup dialog box. Specify the Authentication Server host name and port number and click **Save**.

3. At the third and final configuration dialog box, you can edit Authentication Server information, configure databases and edit the fetch size for data fetched between SAS Federation Server and the web application server.

4. When you are finished with configuration, click **Log Off** in the upper right corner. You will be redirected to the main login page for SAS Federation Server Manager.

Update JVM

SAS Federation Server Manager uses Jetty, an opened source Java application web server. It is best practice to update the following JVM settings to suit your environment:

Note: When making changes to fsmanager.cfg it is best to work with a text editor that can handle the Unicode character set. Some plain text editors might convert characters which can corrupt the file.

-Xms (initial heap size)

-Xmx (max heap size)

-XX MaxPermSize

-Xss (stack size)

To update JVM settings for the Jetty Web Server:

1. Open the **`fsmanager.cfg`** file located in **`SASHome\FSManager\version\etc`**.
2. Use the **`java_args`** option to specify JVM settings.

```
# java_args = <arg1> [<arg2>...]
# Additional arguments for the Java executable used to run the web application.
#
# example: java_args = -Xss1024k -Xmx512m
```

3. Save the **`fsmanager.cfg`** file.
4. Run the **`Refresh Service Options`** script located in the **`FSManager`** directory of the installation root. The JVM settings that are specified in the **`fsmanager.cfg`** file will be applied to Jetty.

Federation Server Manager Service

Introduction

SAS Federation Server Manager runs as a Microsoft Windows service called Federation Server Manager. You can start and stop the service using the Microsoft Management Console (MMC) or through **Administrative Tools** in the **Control Panel**.

In UNIX, SAS Federation Server Manager runs as a daemon administered from a command line using **`fsmadmin`** to start and stop the daemon.

The Federation Server Manager Service in Windows

Start and stop the service using the MMC or the **Control Panel** ⇨ **Administrative Tools**.

1. Click **Start** ⇨ **Control Panel**.
2. Double-click **Administrative Tools** ⇨ **Computer Management**.
3. Expand the **Services and Applications** folder.
4. Click **Services**.
5. Click **Federation Server Manager** <*instance*>.
6. Click either **Stop the service** or **Restart the service**.

Modifying the Windows Service Log On

When SAS Federation Server Manager is installed, the Federation Server Manager service is configured to start with the local system account. Because this account can have restrictions, such as accessing network drives, it is suggested that you modify the service properties so the service log on uses an account with the necessary privileges to

access the required network drives and files. For security reasons, you should assign administrative privileges only if necessary.

To modify the SAS Federation Server Manager log on:

1. Select **Control Panel** ⇨ **Administrative Tools**.
2. Double-click **Services**, and select the **Federation Server Manager service**.
3. Click the **Log On** tab and select **This account**. Enter the account and password credentials for a user with administrative privileges.
4. Restart the **Federation Server Manager service**.

The Federation Server Manager Daemon in UNIX

Start and stop the daemon using the **`fsmadmin`** application included in the installation. This application can be run using the command-line command: **`./bin/fsmadmin`*`command`*** from the installation root directory, where ***`command`*** is one of the following options:

Command	Description
start	Starts the SAS Federation Server Manager daemon. For example: **`./bin/fsmadmin start`**
stop	Stops the SAS Federation Server Manager daemon. For example: **`./bin/fsmadmin stop`**
status	Checks the status of the SAS Federation Server Manager daemon.
help	Displays help information for the **`fsmadmin`** command.
version	Displays the version information for **`fsmadmin`**.

Chapter 4

SAS Federation Server Administration

Overview

SAS Federation Server can be managed with the use of various DDL statements and SQL commands or through the SAS Federation Server Manager user interface. SAS Federation Server Manager provides dialog boxes and wizards that guide you through the completion of a task, without requiring that you know the FedSQL commands required to perform the task. If you are not familiar with FedSQL and federation, you might want to use the SAS Federation Server Manager, which is intuitive and easy to use. If you are already familiar with SQL or FedSQL, you might prefer to use the Administration DDL statements presented in Appendix 1 on page 159. Most of the functions performed with administration DDL, can be accomplished using SAS Federation Server Manager.

The SAS Federation Server Database

Overview

The SAS Federation Server database is a transactional database, or system catalog that contains configuration metadata. Configuration metadata includes the list of created data services, DSNs, privileges, and other information generated as a result of configuring the Federation Server. This information is stored in a Federation Server Database since the metadata must always be in a consistent state and thus requires the use of ACID transactions (atomicity, consistency, isolation and durability).

The system catalog, SYSCAT, contains information about the configuration of SAS Federation Server. This information can be returned to the user through queries against information views on page 188.

Creation of the SAS Federation Server Database

The SAS Federation Server database is created once when SAS Federation Server is initially invoked. At that time, a set of system tables is created to hold various objects that are created as the server is configured. For example, when a data service is created, system tables will be updated to hold the definition of the new data service. Each time a change is made to the server configuration, the system tables in the database are modified. The Database can be backed up at any time to capture and preserve a particular server configuration. The default location of the database is **<installation root>\SASHome\SASFederationServer*version*\var** and is called **`SYSCAT.TDB`**. The name and location can be modified in the **`dfs_serv_common.xml`** configuration file. Other configuration files are affected if the location of the database changes.

If you need to change the location of the SAS Federation Server database, there are three files to modify:

`dfs_serv_common.xml`
Change the value for the TranPath option to point to the new location. This changes the location for the SYSCAT transactional database.

`dfs_log.xml`
Change the value of the DFS_DBAPPENDER_DB entity to point to the new location. This changes the location of the SQL_LOG transactional database.

dfs_log_SQL_Logging.xml
Change the value for the FIREBIRD_LOCK environment variable to point to the new location. This changes the location for transactional database lock files for both the SQL_LOG and SYSCAT transactional databases.

Note: It is a best practice to periodically back up the system catalog database, SYSCAT.TDB.

SAS Federation Server Resource Cache

Overview

Data that is used frequently can be cached from Authentication Server and retained on SAS Federation Server until the cached information is refreshed or purged. This data cache can help improve server performance at both ends by reducing the number of calls needed from SAS Federation Server to Authentication Server.

Managing Named Server Caches

SAS Federation Server maintains several internal resource caches, all of which are designed to improve the performance of potentially expensive operations. An administrative user can manage common cache properties by name by using the ALTER SERVER DDL statement. Among the cached resources are user and group identity information. This information is required in authorization enforcement and multi-tiered authentication, privilege information, and result sets generated from the execution of definer's rights views.

SAS Federation Server can cache resources that are related to authentication, reducing roundtrips to Authentication Server. Several of these configurable caches are periodically repopulated as SAS Federation Server captures information from Authentication Server during the authentication process. The cache names are prefixed with **AS**. By default, resources related to Authentication Server are not cached.

SAS Federation Server can also cache privilege information, reducing internal queries to various system tables related to privileges, thereby improving the rendering of authorization enforcement decisions. The authorization cache is periodically updated as SAS Federation Server performs authorization enforcement and processes DDL such as GRANT, DENY, REVOKE, and various DROP commands. The authorization cache is named **Authorization** and is configured at maximum level by default.

SAS Federation Server can cache result sets of definer’s rights views, improving query execution and data access performance. For information about enabling caching, see Managing Cache Configuration Properties on page 46 .

The following cache namespace table describes the information cached under each name.

Cache Name	Description
AS	All Authentication Server (AS) cached resources
AS.Name	Name to identifier mappings

Cache Name	Description
AS.Name.Subjects	User name to Authentication Server identifier cache
AS.Name.Groups	Group name to Authentication Server identifier cache
AS.Subject	Per user cache resources
AS.Subject.Groups	User group memberships cache
AS.Subject.Principals	User owned principals cache
AS.List	Directory listings
AS.List.Subjects	User listings cache
AS.List.Groups	Group listings cache
Authorization	Privileges cache
ResultSet	Result Sets
ResultSet.View	View result sets cache

Note: SAS Federation Server Manager does not display these values. To view them, use the SQL Console to select from the Information Views—for example, **`SELECT * FROM CONFIG_DATA_SERVICES WHERE DATA_SERVICE_NAME= '__SERVER__'`**.

Managing Cache Configuration Properties

Common cache management operations are handled using the ALTER SERVER command with CACHE list-valued options. This CACHE list-valued option is keyed by the NAME option (similar to the CONOPTS list-valued option, keyed by DRIVER). Values of the NAME option must be one of the names listed in the preceding table.

This statement resets, drops, or adds individual properties of the named cache:

```
ALTER SERVER {OPTIONS( cache-option-list [,cache-option-list ...] )}
cache-option-list ::= CACHE( NAME cache-name , cache-properties )
```

This statement drops properties currently persisted with the named cache and reverts their run-time settings to defaults:

```
cache-option-list ::= DROP CACHE( NAME cache-name )
```

This statement resets or adds properties of the named cache as a complete set, replacing any existing properties:

```
cache-option-list ::= SET|ADD|XSET CACHE( NAME cache-name, cache-properties )
```

The NAME option is required and specifies the name of the cache to be managed. Properties of the cache are replaced or created within the sublist. Normal generic SQL options syntax applies to the cache option and the associated suboptions outlined in Cache Properties.

Cache Properties

TIMEOUT *timeout*

The timeout value specifies the length of time in seconds a resource may be cached before being considered stale and marked for on-demand refresh. Once a resource becomes stale, it is typically refreshed and reached on its next access. All caches support the TIMEOUT option. Here are the default timeout values associated with each of the caches:

Name	Default Timeout Value
ResultSet	1800 (30 minutes)
ResultSet.View	1800 (30 minutes)
Authorization	-1 (infinite)
All others	0 (not applicable — not cached)

The TIMEOUT property can be restored to a default several ways once it is explicitly configured. For example, in the following scenario, the configured timeout values for result set caching are as follows:

ResultSet = 3600 (1 hours)

ResultSet.View = 3600

The following statement overrides both of these timeout values:

```
ALTER SERVER {options cache(name ResultSet, xset timeout 300)}
```

The statement sets the timeout of ResultSet to 300 seconds explicitly and also sets all children (for example, ResultSet.View) to 300 seconds. Note that the statement persists only the new timeout value for the cache (ResultSet), but changes the current value for all the children (ResultSet.View) as well. This allows top-down run-time management of timeout values while preserving the configured defaults of child names.

To reset the timeout to the original default value, issue the TIMEOUT option with no value:

```
ALTER SERVER {options cache(name ResultSet.View, xset timeout)}
```

Cache properties are inherited from the parent namespace when the cache configuration is dropped altogether:

```
ALTER SERVER {options drop cache(name ResultSet.View)}
```

Afterward, the ResultSet.View cache inherits the timeout value from the parent namespace (ResultSet), which is 300 seconds.

PURGE | **FLUSH**

Specifies that the cache named should be refreshed. Associated resources will be reacquired and cached on next access and can be flushed immediately. This option is not persisted, and its use does not affect existing properties already configured for the cache named. All caches support the FLUSH option.

LEVEL *level*

Controls the caching granularity of the named cache. This property applies to the Authorization cache only. Valid values are as follows:

ALL / OBJECT	Cache privileges for columns, tables and all higher level securables. This is the default privilege caching level.
CONTAINER	Cache privileges for schemas and all higher level securables.
NONE / OFF	Turn off all privilege caching.

Connection Pooling

Overview

Connection pooling is a reserve of database connections that are maintained in SAS Federation Server so that the connections can be reused as future requests to the database are required. Opening and maintaining a database connection for each user, especially requests made dynamically, is costly and resource intensive. With connection pooling, connections are created and placed into the pool to be used over again so that a new connection to a back-end data source does not have to be reestablished. This practice reduces the amount of time it takes to establish a connection to a database. If all the pooled connections are in use, and the pool is large enough to hold a new connection, then a new connection is made and added to the pool.

If connection pooling is enabled, the client connects to a data source as usual. If there is an existing database connection in the connection pool that meets the client's requirements (for example, a connection to the desired database using the applicable credentials), then that connection will be used by the client. Otherwise, a new connection is created.

When the client disconnects, the server evaluates whether to keep the underlying connection in the connection pool, or whether to free it. If the connection will not be pooled, then the connection is freed when the client frees its connection handle.

Options

The following options control connection pooling on the server. The options are controlled by the ALTER SERVER DDL statement on page 159 .

Enable Connection Pooling

`CONNECTION_POOLING`
`[N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1]`

This option controls whether connection pooling is enabled or disabled for the server. If connection pooling is switched on, connections to databases are not disconnected immediately when the client requests to disconnect from the database. The connections are put into a pool of connections that can be reused by subsequent requests to connect to the same database with the same attributes and credentials. Connections used for Memory Data Store cannot be pooled.

Connection Pool Timeout

`CONNECTION_POOL_TIMEOUT`
`seconds`

This option identifies the time in seconds an unused connection stays in the connection pool. The default is 60 seconds. If the time is exceeded, the connection is removed from the pool and the connection is closed. A value of -1 indicates that the connection never times out and can stay in the pool indefinitely. These connections are freed when the server is stopped.

Maximum Unused Connections

`CONNECTION_POOL_MAXSIZE` *maxsize*

This option identifies the maximum number of unused connections in the connection pool. The default is 50. If the maximum number of connections is reached and a new connection is added to the connection pool, the oldest connection is removed from the pool and that connection is closed. If this option is set at 0, the default of 50 is used. Connections used for Memory Data Store cannot be pooled.

Drop Connection Pooling

`DROP CONNECTION_POOLING`

This option drops and also disables the connection pooling option.

Drop Connection Pool Timeout Option

`DROP CONNECTION_POOL_TIMEOUT`

This option removes connection pool timeout value. If connection pooling is enabled and connection pool timeout option has been dropped, it will use the default timeout value, 60.

Drop Connection Pool Maxsize Option

`DROP CONNECTION_POOL_MAXSZIZE`

This option removes connection pool maxsize value. If connection pooling is enabled and connection pool maxsize option has been dropped, it will use the default maxsize value, 50.

SQL Logging

Overview

SQL Logging is the ability to view SQL statements submitted to SAS Federation Server, combined with other information such as the user who submitted the SQL and a breakdown of the SQL into prepare, execute and cursor phases. Metrics are also available, including elapsed time, number of rows fetched, and size of data moved (fetched or inserted). SQL Logging provides critical information for all server activity, so it can easily be determined who is accessing the system, when they were connected, and what work they performed.

Errors and informational messages that occur when writing SQL Log records to the EVENTS table are recorded in the server's log and appear with a prefix of **`DBAppender<SQL_LOG>:Append.`** See the "Server Logging Configuration" on page 59 topic for further information on the logging facility for SAS Federation Server.

Configuring SQL_Logging

Introduction

Configuration of SQL Logging is facilitated using the **dfs_log_SQL_Logging.xml** located in the **/etc** directory of the Federation Server installation path.

Enable SQL Logging

To enable SQL Logging, open **dfs_log_SQL_Logging.xml** for editing and set the top-level logger to TRACE. There is no need to change any of the configuration parameters. The logging level value should always be set at TRACE so that all information is captured.

Following top-level logging are transactions that you can use to control SQL Logging detail. Remove the comment marks from each line to display logging details:

Enable SQL Logging in SAS Federation Server Manager

SAS Federation Server always starts with SQL Logging set at the default level in the configuration file, but can be enabled or disabled dynamically for a server session. This is done through the SQL Log tab of the Federation Server Properties dialog box. When SQL Logging is modified from the SQL Log tab, it is active for the server session only. When the server is restarted, the level of logging reverts to the value specified in the SQL logging configuration file.

Note: Configuring SQL Logging in SAS Federation Server Manager activates logging for the server session only.

To enable SQL Logging in SAS Federation Server Manager:

1. Select a Federation Server in the tree.
2. Open the action menu in the upper left corner and select **Properties**.
3. The Federation Server Properties window appears. Click the **SQL Log** tab.
4. Click to select **On. Log SESSION** and select the events to record.

Configuring a 3rd Party DBMS for SQL Logging

If you are using a database other than the SQL_LOG database that installs with SAS Federation Server, set up your database according to vendor specifications. SAS Federation Server ships with example configuration files for a few select databases such as Oracle and DB2. These files are located in the **/etc** directory of the Federation Server installation path. After setting up your database and specifying the domain, configure a connection to the driver, and modify the data types to suit your environment as outlined below.

Change the Domain

Using the ALTER DATA SERVICE command, change the domain for SQL_LOG to the domain that was created for the data service. Here is an example:

```
ALTER DATA SERVICE SQL_LOG domain ORA1
```

Modify the Connection String

Modify the connection string to include the values for your SQL Logging database. For example,

```
param name="connectionString"
value="CATALOG=*;DEFAULT_CATALOG=catalog_name;DRIVER=driver;
CONOPTS=(DSN={yourdsn};UID='your user id';PWD='your password')"
```

For a list of possible connection options see the "Database Functionality and Driver Performance" on page 110 in this guide.

Create the Table / Modify Data Types

At the end of the CREATE TABLE statement, update **'IN <name of your table space>'** to reflect the value for your setup.

Modify the columns that appear in the SQL Logging configuration file to map to data types supported by the new database. A complete list of SQL Logging columns and data types appears in "The EVENTS Table" on page 53 .

```
CREATE TABLE &DFS_DBAPPENDER_TABLE;
    (ARM_NAMESPACE VARCHAR(256),
    TRAN_TIMESTAMP TIMESTAMP,
    APP_HANDLE VARCHAR(36),
    APP_ID VARCHAR(36),
    APP_NAME VARCHAR(50),...
    ) IN <name of your table space>"/>
```

Note: For additional setup information, refer to the sample configuration files located in the **/etc** directory of the Federation Server installation path.

ARM Transactions

The table below shows the ARM transactions that are captured in SAS Federation Server. With SQL Logging enabled, information in each of the transactions is captured. Following the table is a brief explanation of each of the transactions.

Name	Type	Associated Namespace
SESSION	Session transaction	Perf.ARM.FederationServer.Session.Transaction.SESSION
DBC	Database connection	Perf.ARM.SQLServices.Connection.Transaction.DBC
DBTRAN	Database transaction	Perf.ARM.SQLServices.Connection.Transaction.DBTRAN
SQL	SQL statement	Perf.ARM.SQLServices.Statement.Transaction.SQL
Prepare	SQL Statement	Perf.ARM.SQLServices.Statement.Prepare
Execute	SQL Statement	Perf.ARM.SQLServices.Statement.Execute
CURSOR	SQL Statement	Perf.ARM.SQLServices.Statement.Transaction.CURSOR
Fetch Scroll	SQL Statement	Perf.ARM.SQLServices.Statement.FetchScroll
SetPos	SQL Statement	Perf.ARM.SQLServices.Statement.SetPos
BulkOps	SQL Statement	Perf.ARM.SQLServices.Statement.BulkOperations
Fetch	SQL Statement	Perf.ARM.SQLServices.Statement.Fetch

- **SESSION**: (Session Transaction) A session transaction starts when a user initiates a server session.
- **DBC**: (Database Connection) A database connection transaction is a child object of the SESSION transaction. A database connection begins when a user connects to a data source and ends when the user disconnects from the data source.
- **DBTRAN**: (RDBMS Transaction) DBTRAN is the actual database transaction. It is a child object of the DBC transaction. A DBTRAN transaction begins with an established driver connection, or when a previous transaction is committed or rolled back, such that a new one begins. DBTRAN records are written to the log only if AUTOCOMMIT is set to OFF. The DBTRAN transaction stops when AUTOCOMMIT is set to ON or when a COMMIT or ROLLBACK command is issued. SQL statements can span DBTRAN transaction boundaries.
- **SQL**: (SQL Statement) SQL is a logical transaction. It encapsulates a series of activities related to one SQL statement. It is a child object of a DBC transaction. An SQL transaction starts when a user issues an SQL statement. Regardless of the statement type (DQL, DML or DDL) the SQL transaction stops when the statement is either closed, or the call to Prepare ends. Subsequent executions of the same statement are recorded under the same SQL transaction, even if the statement is a DQL and the result set associated with it has been is closed.
- **Prepare**: The prepare transaction measures the Prepare phase of an SQL statement. It is a child object of an SQL transaction. The Prepare transaction starts when a user Prepares an SQL statement and stops when the call to prepare returns.
- **Execute**: The execute transaction measures the Execute phase of an SQL statement. It is a child object of the SQL transaction. The EXEC transaction starts when a user executes an SQL statement and stops when the call to execute returns.
- **CURSOR**: CURSOR is a logical transaction. CURSOR is a child object of an SQL transaction and it encapsulates all operations executed in a cursor, including reading, positioning and updates. The CURSOR transaction starts when the Execute transaction finishes. It stops when the cursor is closed. All operations on the same result set belong to the same CURSOR transaction.
- **Fetch/Fetch Scroll**: The FETCH transaction is a child object of the CURSOR transaction. The FETCH transaction has an Execute transaction as its predecessor. It is started when a user issues the first fetch on a result set using Fetch or Fetch Scroll. It stops when the call to Fetch or Fetch Scroll returns.
- **SetPos**: The SetPos transaction is a child object to a CURSOR transaction. The SetPos transaction has an execute transaction as its predecessor. It is started when a user issues a SetPos call and stops when the call returns.
- **BulkOperations** : The BulkOperations transaction is a child object of a CURSOR transaction. The BulkOperations transaction has an Execute transaction as its predecessor. It is started when a user issues a call to BulkOperations and stops when the call returns.

The SQL_LOG Data Service and DSN

Whether SQL Logging is enabled or disabled on SAS Federation Server, the default configuration will automatically create an SQL_LOG data service and DSN. When the data service is created, the server also creates an SQL_LOG transactional database and creates an EVENTS table within the database. This table contains data captured for specific activity in the server, such as information about SQL statements submitted by connected users. The data service, DSN, database and table are always created so that

they are available, even if the server is not initially invoked with SQL Logging enabled. As noted above, SQL Logging can be enabled or disabled dynamically at any time, so the server ensures that the SQL Logging constructs are always created when the server starts.

When connected to the SQL_LOG data service and DSN:

- Catalog functions are restricted so that they only return the SQL_LOG table. Therefore, only the SQL_LOG catalog, schema, and table are visible.
- Privileges will restrict access to anything other than the SQL_LOG table, therefore, only the SQL_LOG catalog, schema and table are visible.
- The administrator can assign CONNECT privilege on the SQL_LOG DSN. Federation Server SQL Authorization Enforcement is enabled by default.
- The administrator can assign CONNECT, SELECT, and DELETE privileges only.
- You cannot create new tables or insert into tables through the SQL_LOG data service.

The valid privileges are CONNECT, SELECT and DELETE.

- The SQL_LOG data service will allow SELECT or DELETE privileges for the active SQL Logging table and associated columns.
- CONNECT is valid on the SQL_LOG data service, catalog, and DSN.

Use GRANT, REVOKE, or DENY to set privileges for the SQL_LOG data service, DSN, catalog, schema, table, and associated columns.

The EVENTS Table

About the EVENTS Table

The EVENTS table resides in the SQL_LOG database that is created with the SQL_LOG data service as a result of enabling SQL Logging. The EVENTS table contains transactional data records written from SAS Federation Server.

Data captured and stored in the EVENTS table includes the number of bytes inserted from an SQL transaction. This does not include any literal data sent to SAS Federation Server. Only data sent through bound memory buffers, such as parameter data, is included. Also, the number of bytes inserted reflects the amount of data stored in the bound memory locations. It does not reflect the size of the data on disk.

Managing the EVENTS Table

The EVENTS table must be managed manually. With SQL Logging enabled, data records are continuously written to the EVENTS table. Therefore, the table increases in size when the server is active and processing requests. SAS Federation Server maintains logging data indefinitely until the table is managed or purged. You can use the SQL console window to issue commands to manage the EVENTS table. By connecting with the SQL_LOG DSN which has FedSQL dialect enabled, use any SELECT or DELETE statement that is supported by the FedSQL language to manage the table.

CAUTION:

Do not change the name of the EVENTS table.

Determine the Size of the EVENTS Table

Because the size of the EVENTS table increases as activity is logged, you should check the size of the table periodically to determine if records need to be archived or deleted. Use the following statement to calculate the amount of rows in the EVENTS table:

```
SELECT COUNT(*) FROM EVENTS
```

Archiving Data in the EVENTS Table

As the EVENTS table grows in size, you can move data to another table for archive purposes. This is accomplished by creating a federated DSN to the SQL_LOG DSN and another data source to use for storing the archived data.

- Use the following statement to move data to a new table:

```
CREATE TABLE
<archive_catalog>.<archive_schema>.<archive_table> AS SELECT *
FROM SQL_LOG.SQL_LOG.EVENTS WHERE <where_condition>
```

- To determine what records are needed, use the WHERE condition with dates:

```
WHERE TRAN_TIMESTAMP > TIMESTAMP '2012-01-25 01:00:00'
```

- To move data to an existing table, use the INSERT statement:

```
INSERT INTO
<archive_catalog>.<archive_schema>.<archive_table> SELECT *
FROM SQL_LOG.SQL_LOG.EVENTS WHERE <where_condition>
```

Deleting Data in the EVENTS Table

Use the DELETE statement to remove outdated records that are no longer needed or have been archived. Here is an example:

```
DELETE FROM SQL_LOG.SQL_LOG.EVENTS
WHERE TRAN_TIMESTAMP < TIMESTAMP '2011-04-11 12:00:00'
```

Columns and Data Types

The following table presents the columns and associated data types that reside in the EVENTS table.

Table 4.1 *Column and Data Types of the EVENTS Table*

Column	Data Type	Description
APP_HANDLE	VARCHAR(36)	A unique ID that is associated with an application instance.
APP_ID	VARCHAR(36)	The application ID.
APP_NAME	VARCHAR(50)	The application name.
ARM_NAMESPACE	VARCHAR(256)	The logger name (namespace) of the logging event.
AUTHORIZATION_ID	VARCHAR(128)	UUID for the user from the Authentication Server.
AUTHORIZATION_NAME	VARCHAR(128)	The user name from the Authentication Server.

Column	Data Type	Description
BYTES_FETCHED	BIGINT	The size of data read in bytes.
BYTES_INSERTED	BIGINT	The size of data inserted in bytes.
CACHE_VIEW_CATALOG	VARCHAR(256)	The cache view catalog name.
CACHE_VIEW_NAME	VARCHAR(256)	The cache view name.
CACHE_VIEW_SCHEMA	VARCHAR(256)	The cache view schema name.
CATALOG_NAME	VARCHAR(256)	The catalog name.
CLIENT_CORRELATOR	VARCHAR(56)	The transaction's client correlator (base64 encoded).
CONNECTION_DRIVER	VARCHAR(25)	The driver that was used for the connection. For example, FEDSQL, ORACLE, TERADATA, ODBC, MYSQL, DB2, and others.
CONNECTION_NAME	VARCHAR(256)	The expanded connection string.
CONNECTION_TRAN_HANDLE	VARCHAR(36)	The DBC transaction under which the current transaction is assigned to.
CURR_SAS_TIMEOFDAY	DOUBLE PRECISION	The current time-of-day for the ARM event.
CURR_SYSTEM_CPU_TIME	DOUBLE PRECISION	The process current system CPU time for the ARM event.
CURR_USER_CPU_TIME	DOUBLE PRECISION	The process current user CPU time for the ARM event.
CURRENT_CORRELATOR	VARCHAR(56)	The transaction's correlator (base64 encoded).
CURSOR_TRAN_HANDLE	VARCHAR(36)	The CURSOR transaction under which the current transaction is assigned to.
DBTRAN_STATE	VARCHAR(15)	The state of the current transaction. For example, OPEN, CLOSED.COMMIT, CLOSED.ROLLBACK.
DBTRAN_TRAN_HANDLE	VARCHAR(36)	The UUID that points to the driver's DBTRAN transaction handle, which is not its parent (the CONNECTION handle is parent) since the SQL can span multiple DBMS transactions.
EVENT_SEQUENCE	BIGINT	A unique value associated with the ARM record. Values increase for each record, usually with an increment of 1 (database-specific).
EXEC_PARAM_DATA	VARCHAR(1024)	The XML format for encoding parameter array data.

Column	Data Type	Description
GROUP_NAME	VARCHAR(128)	The group name of the application instances.
IO_COUNT	BIGINT	The total number of process disk, tape or related input and output operations for the transaction event.
IP_ADDRESS	VARCHAR(48)	The IP address of the client.
LOGIN_ID	VARCHAR(128)	The current user ID that is associated with the transaction.
MEM_CURRENT	BIGINT	The current process memory utilization for the transaction event.
MEM_HIGH	BIGINT	The highest amount of process memory used for the transaction event.
OBJECT_NAME	VARCHAR(256)	The object name used in the SQL statement.
OBJECT_TYPE	VARCHAR(60)	The type of object that was accessed.
PARENT_CORRELATOR	VARCHAR(56)	The transaction's parent correlator (base64 encoded).
PREDECESSOR_TRAN_HANDLE	VARCHAR(36)	The driver's Execute transaction handle. This ties the Fetch transaction to an execution and its metrics.
ROWS_DELETED	BIGINT	The number of rows deleted.
ROWS_FETCHED	BIGINT	The number or rows read.
ROWS_INSERTED	BIGINT	The number of rows inserted.
ROWS_UPDATED	BIGINT	The number of rows updated.
SCHEMA_NAME	VARCHAR(256)	The name of the schema being accessed.
SERVER_MESSAGE	VARCHAR(500)	The message associated with the event.
SESSION_TRAN_HANDLE	VARCHAR(36)	The SESSION transaction that the current transaction is assigned to.
SOURCE_FILE_NAME	VARCHAR(128)	The file name where the logging request was issued.
SQL_DIALECT	VARCHAR(15)	The dialect that is being used: FEDSQL or NATIVE.
SQL_TRAN_HANDLE	VARCHAR(36)	The SQL transaction that the current transaction is assigned to.

Column	Data Type	Description
STATEMENT_ID	BIGINT	The SQL statement hash. The value derived from the SQL statement content.
STATEMENT_NAME	VARCHAR(256)	The SQL statement name.
STATEMENT_PLAN	VARCHAR(15000)	Column valued for SQL statement types only. *Note:* The plan value may truncate if the character limit is exceeded. For example, Oracle's VARCHAR limit is 4000 while SQL Server is 8000.
STATEMENT_STATE	VARCHAR(15)	The state of the SQL statement. For example, S0, S1 and S2.
STATEMENT_TEXT	VARCHAR(15000)	The text of the SQL statement.
STATEMENT_TYPE	VARCHAR(15)	The type of SQL statement, such as DQL, DQL.Metadata (catalog methods), DML, or DDL. Empty if unknown.
THREAD_CURRENT	BIGINT	The current process thread count for the transaction event.
THREAD_HIGH	BIGINT	The process highest thread count for the transaction event.
TRAN_CLASS_ID	VARCHAR(36)	The UUID of transaction class.
TRAN_HANDLE	VARCHAR(36)	The UUID of transaction instance.
TRAN_NAME	VARCHAR(50)	SESSION, DBC, DBTRAN, SQL, Execute, Fetch, CURSOR. For additional information see “ARM Transactions ” on page 51.
TRAN_START_SAS_TIMEOFDAY	DOUBLE PRECISION	The time-of-day value for the current transaction start event.
TRAN_STATE	VARCHAR(15)	The state of the transaction: START, STOP, UPDATE, BLOCK, UNBLOCK, DISCARD.
TRAN_STATUS	VARCHAR(15)	The transaction status: UNKNOWN, ABORTED, GOOD, FAILED, STOP.
TRAN_STOP_SAS_TIMEOFDAY	DOUBLE PRECISION	The time-of-day value for the current transaction stop event.
TRAN_TIMESTAMP	TIMESTAMP	The current timestamp of the transaction.
TRANRESP_SYS_CPU_TIME	DOUBLE PRECISION	The calculated system CPU time for the duration of the transaction.

Column	Data Type	Description
TRANRESP_TIME	DOUBLE PRECISION	The calculated elapsed time for the duration of the transaction.
TRANRESP_USER_CPU_TIME	DOUBLE PRECISION	The calculated user CPU time for the duration of the transaction.
TRANSTART_SYS_CPU_TIME	DOUBLE PRECISION	The process system CPU time for the current transaction start event.
TRANSTART_USER_CPU_TIME	DOUBLE PRECISION	The process user CPU time for the current transaction start event.
TRANSTOP_SYS_CPU_TIME	DOUBLE PRECISION	The process system CPU time for the current transaction stop event.
TRANSTOP_USER_CPU	DOUBLE PRECISION	The process user CPU time for the current transaction stop event.

SQL Logging Performance Tuning

Introduction

You can configure SQL Logging so that it will have minimum impact on SAS Federation Server performance, especially during periods of heavy processing.

Using Maximum Buffered Events (MaxBufferedEvents)

Maximum buffered events tells the server how many events to buffer before writing them to the EVENTS database. You can set this option using the MaxBufferedEvents option in the SQL Logging configuration file. The default setting for MaxBufferedEvents is 100 meaning that 100 events are buffered before writing them to the EVENTS table. Setting MaxBufferedEvents to a higher number might consume server resources but will show improved performance during periods of heavy processing. A sustainable high value for MaxBufferedEvents is approximately 500. Here is the syntax used in the SQL Logging configuration file:

```
<param name="maxBufferedEvents" value="100" />
```

Configure Indexing on the EVENTS Table

When using the default configuration of the TRAN data store, writes to the SQL Logging database can sometimes affect overall performance, especially during periods of heavy processing. This is caused by the indexes that are present on the EVENTS table. During these high activity periods, you can drop indexes on the EVENTS table by updating the SQL_LOG data service as shown here:

```
alter service SQL_LOG {options AUTOINDEX off}
```

To activate indexing on the EVENTS table:

```
alter service SQL_LOG {options AUTOINDEX on}
```

Because indexes are required to process SQL queries in SAS Federation Server Manager, it is not advisable to view SQL Logging information using SAS Federation

Server Manager during periods when indexes are not active. You can re-enable indexes after processing activity decreases on SAS Federation Server. At that time you should be able to view SQL Logging information in SAS Federation Server Manager.

An alternative to dropping indexes is to configure SQL Logging to use an external, or third party, database. For more information see "Configuring a 3rd Party DBMS for SQL Logging" on page 50.

Server Logging Configuration

Introduction

The logging facility is a flexible, configurable framework used for collecting, categorizing, and filtering events that are generated by processes and writing events to a variety of output devices. The **dfs_log.xml** configuration file controls the destination, contents, and formats of the logging facility log for SAS Federation Server. The configuration file specifies options for loggers, appenders, levels, filters, and the layout of log messages. You can change logging levels dynamically without stopping the server.

Initial Logging Configuration

The default logging facility configuration for SAS Federation Server includes a definition for the RollingFileAppender. The appender routes events to a rolling log file.

The rolling log file is configured as follows:

- A new log is created when the date changes and when a new server process is started.
- Events are written by using a layout that includes the current date, current time, logging level, process ID, the user identity that is associated with the event, and a message.
- The name of the rolling log file follows the following convention:

  ```
  dfs_%d_%S{pid}.log
  ```

  where **%d** is the date and **%S{pid}** is the process ID number (PID) for SAS Federation Server.
- The rolling log files are placed in the **/var/log**directory.
- When a new rolling log file is created, a heading is written to the file. The heading identifies the server's host machine, operating system, operating system release number, and server start-up command.

The following table lists the loggers that reference the RollingFileAppender:

Logger Name	Logging Level	Description
Admin	Info	processes log events that are relevant to system administrators or computer operators.

Logger Name	Logging Level	Description
App	Info	processes log events that are related to specific applications. For example, metadata servers, OLAP servers, stored process servers, and workspace servers use loggers that are named App.class.interface.method to record method calls that are issued to the server.
App.Server	Info	Server top-level object runtime and interface events.
	Debug	Method call and return events.
	Trace	Method parameters.
App.Session	Info	Session object runtime and interface events.
	Debug	Method call and return events.
	Trace	Method parameters.
App.Connection	Info	Connection object runtime and interface events
	Debug	Method call and return events.
	Trace	Method parameters.
App.Statement	Info	Statement object runtime and interface events
	Debug	Method call and return events.
	Trace	Method parameters.
App.Program	Info	General application independent events including errors and warnings from arbitrary services or the OS
Audit	Info	Processes log events to be used for auditing. These events include updates to public metadata objects, user access to SAS libraries, accepted and rejected user authentication requests, and administration of users, groups, and access controls.
Audit Authentication	Info	Authentication provider events.
Audit Table	Info	Federation server specific events.
Audit Table Connection	Info	Audit events related to server connections, including connection pooling and dynamic connections.
Audit.Table.Security	Info	Federation Server authorization events.
Audit.Table.Security.Provider	Info	Detailed authorization services runtime events relating to user identity management and access control logic and enforcement decisions.

Logger Name	Logging Level	Description
Logging	Error	Log4SAS configuration and runtime events.
Logging.Appender	Error	Appender specific configuration and runtime events.
Logging.Appender.DB	Error	DB Appender specific events (used in SQL Logging).
Cradle	Info	General server process framework services, startup and termination events.
IOM	Info	Processes log events for servers that use the Integrated Object Model (IOM). The IOM interface provides access to Foundation SAS features such as the SAS language, SAS libraries, the server file system, results content, and formatting services. IOM servers include metadata servers, OLAP servers, stored process servers, and workspace servers.
IOM.Proxy	Info	Server to server outcall events.
	Debug	Method call and return events.
	Trace	Method parameters.
IOM.PE	Warn	Communication protocol engine events.
Perf	Error	Processes log events that are related to system performance.
Perf.ARM	Error	Application Response Measurement performance events.
Perf.ARM.IOM.Session	Error	Session API performance events.
Perf.ARM.IOM.Environment	Error	Environment API performance events.
Perf.ARM.IOM.Connection	Error	Connection API performance events.
Perf.ARM.IOM.Statement	Error	Statement API performance events.
Perf.ARM.FederationServer	Warn	Federation server API independent performance events.
Perf.ARM.SQLServices	Warn	Local SQL services performance events.
<root>	Error	All events produced from SAS® Federation Server.

Logging Thresholds

The SAS logging facility provides six thresholds: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL. Thresholds are used to ignore log events that are lower than a particular level, or to filter messages so that only a single message level is logged.

When a log event occurs, up to three levels of filtering can take place:

1. filtering log events by comparing the log event level to the log event's logger level.
2. filtering log events by comparing the log event level to the appender's threshold.
3. filtering log events by comparing the log event level to the threshold that is specified in the filter definition, which is a part of the appender configuration.

In the first two cases, if the log event level is lower than the logger or appender threshold, the logging facility ignores the log event. Otherwise, processing of the log event continues.

In the third case, the log event level is compared to the filter threshold. If there is a match, the log event can be either accepted or denied. If there is no match, the filtering process continues to the next filter in the filtering policy.

The logging levels, from the lowest to the highest, are as follows:

Level	Description
TRACE	Produces the most detailed information about your application. This level is primarily used by SAS Technical Support or development.
DEBUG	Produces detailed information that you use to debug your application. This level is primarily used by SAS Technical Support or development.
INFO	Provides information that highlights the progress of an application.
WARN	Provides messages that identify potentially harmful situations.
ERROR	Provides messages that indicate that errors have occurred. The application might continue to run.
FATAL	Provides messages that indicate that severe errors have occurred. These errors will probably cause the application to end.

Note: The logging level must be enclosed in quotation marks.

By default appenders do not have a threshold but a threshold can be configured. When set, all log events that have a level lower than the threshold are ignored by the appender.

Modifying the Server Logging Configuration

You can modify the logging facility configuration for SAS Federation Server by modifying the dfs_log.xml file located in the /etc directory of the installation path. Before modifying the file, be sure to make a backup copy.

Here are some examples of changes that you might want to make:

- configure RollingFileAppender to use a different log filename, to roll over log files more or less frequently, or to roll over log files based on file size rather than date.
- specify additional appenders.
- use filters to limit the events that are written to an appender.
- configure a different message layout for an appender.

Trace Log

By tracing each internal API routine that is called by the application, a trace log records transactions that can be used for debugging connection and processing issues. For example, you can request information that traces the FedSQL statements that are submitted to a data service.

Note: By default, tracing is not activated for server logging. You should not activate tracing unless you are instructed to do so by Technical Support.

Tracing can be activated by using the following methods:

- connection string options
- server start-up options
- data service connection arguments
- system options

When you activate tracing, you also specify the physical location where the transaction records are saved. Because SAS Federation Server supports one root file trace log directory and multiple subdirectories, you can group trace logs if necessary.

Utilities for SAS Federation Server

Introduction

SAS Federation Server contains several utilities that assist with management of your server environment. Utilities include database migration, password encryption and system database backup and restore. Utilities are available for the UNIX and Windows operating systems.

UNIX Utilities

dfsadmin Utility

SAS Federation Server for UNIX contains the **`dfsadmin`** utility, located in the **`bin`** directory of the installation root. Run any of the commands in **`dfsadmin`** using the following syntax: **`./bin/dfsadmin`** ***`yourcommand`*** where *yourcommand* is one of the following commands:

Command	Description
`start`	Starts SAS Federation Server. Example: **`./bin/dfsadmin start`**
`stop`	Stops SAS Federation Server. Example: **`./bin/dfsadmin stop`**
`status`	Checks the run status of SAS Federation Server.
`migrate`	Migrate the Federation Server database from an older version options: <old_install_dir>

Command	Description
crypt	Print the encrypted version of a password options: [<password>] For additional information see Password Encryption.
help	Displays command usage and options.
version	Displays version information.

Password Encryption

The **dfsadmin** utility contains a **crypt** command from where you can key the password on the command line in plain text, or let it prompt you so that the password is not echoed to terminal. The password is encrypted using the encryption type installed on the server.

For example,

```
./bin/dfsadmin crypt my_passwd
```

Key a password with the **crypt** command:

```
[admin@reason fedserver]$ ./bin/dfsadmin crypt my_passwd
{SAS002}1950043249132FE80CEC515733C508D5
```

Invoke the **crypt** command and let it prompt for a password:

```
[admin@reason fedserver]$ ./bin/dfsadmin crypt
Enter password:
Confirm password:
Encrypted value: {SAS002}1950043249132FE80CEC515733C508D5
```

Note: Enclose the encrypted password in single quotation marks when using it in a connection string.

Database Backup and Restore

It is recommended that you backup the Federation Server databases periodically, especially the system catalog, SYSCAT.tdb. A utility named **dfsutil** is provided with SAS Federation Server installation and is located in **/bin** of the installation directory. Using **dfsutil**, you can back up the system database (SYSCAT.tdb) and other databases such as the SQL Logging database (SQL_Log). However, the SQL Logging database cannot be backed up with **dfsutil** if it was changed to a third party data store. You can also restore databases using **dfsutil** as long as the database was backed up using the same utility.

Note: If you backup a database with **dfsutil**, then you are required to restore that database using **dfsutil**.

Backup

To backup a database, use the **-db** parameter with the **dfsutil** command. For example:

```
install-path/bin/dfsutil
backup -db syscat
/path/to/backup
install-path/bin/dfsutil
backup -db sql_log
/path/to/backup
```

Restore

To restore a database, use the **-db** parameter with the **dfsutil** command. For example:

```
install-path/bin/dfsutil
restore -db syscat
/path/to/backup
install-path/bin/dfsutil
restore -db sql_log
/path/to/backup
```

Windows Utilities

Password Encryption

Use the **dfs_crypt** utility to encrypt passwords. **dfs_crypt**, is located in the **/bin** directory of the installation path for SAS Federation Server.

- Navigate to **[drive:]\Program Files\SASHome\FedServer \server1\bin**> and enter: **dfs_crypt password**
- An encrypted password is returned: **{SAS002}A984D9084D59A9EA**

Note: Enclose the encrypted password in single quotation marks when using it in a connection string.

Database Backup and Restore

It is recommended that you backup the Federation Server databases periodically, especially the system catalog, SYSCAT.tdb. A utility named **dfsutil** is provided with SAS Federation Server installation and is located in **/bin** of the installation directory. Using **dfsutil**, you can back up the system database (SYSCAT.tdb) and other databases such as the SQL Logging database (SQL_Log). However, the SQL Logging database cannot be backed up with **dfsutil** if it was changed to a third party data source. You can also restore databases using **dfsutil** as long as the database was backed up using the same utility.

Note: If you backup a database with **dfsutil**, then you are required to restore that database using **dfsutil**.

Backup

To backup a database, use the **-db** parameter with the **dfsutil** command. For example:

```
install-path\bin\dfsutil
backup -db syscat
\path\to\backup
install-path\bin\dfsutil
backup -db sql_log
\path\to\backup
```

Restore

To restore a database, use the **-db** parameter with the **dfsutil** command. For example:

```
install-path\bin\dfsutil
restore -db syscat
\path\to\backup
install-path\bin\dfsutil
```

```
restore -db sql_log
\path\to\backup
```

Uninstalling SAS Federation Server

The uninstaller for SAS Federation Server removes all directories and configuration files from the installation root which also includes the Federation Server Database which includes the system catalog. Before uninstalling Federation Server, you should first install and configure a new server instance, migrate your database, and then check the previous server instance for any files that should be copied to the new server. If you want to keep any of your configuration files, copy them out of the /etc directory before you run the uninstall process.

The following items are deleted when uninstalling SAS Federation Server:

1. The Federation Server Database called **SYSCAT.TDB** which is located in the **var** directory of the install by default. This is where the system tables are stored.
2. The SQL Logging database called **SQL_LOG.TDB** which is also located in the **var** directory of the install by default. This stores monitoring data for SQL Logging.
3. Configuration files especially if modifications were made after installation. The configuration files are located in the **etc** directory and the **etc/migrate** directory of the installation path. The configuration file names are listed in the table below.
4. Any log files in the **var/log** directory of the install.

Table 4.2 *Federation Server Configuration Files*

Configuration Files in the Directory:	Configuration Files in the etc/migrate Directory:
dfs_log.xml	**dfs_log_213_310.xml**
dfs_log_SQL_Logging.xml	**dfs_serv_213_310.xml**
dfs_serv.xml	**dfs_serv_213_310_sql.xml**
dfs_serv_common.xml	
dfs_entities.dtd	

Chapter 5

SAS Federation Server Security

Overview

Properly configured security for SAS Federation Server ensures that both the server and its data are secure. Data is protected against unauthorized access, and can be guaranteed secure transmission lines for transferring data. The security features are flexible with respect to the types and amount of security, and both security setup and maintenance are easily managed.

Authentication

SAS Federation Server must be configured to use an Authentication Server. When a user connects to SAS Federation Server to access data, the user's authenticating credentials are passed to the Authentication Server for validation. Once the credentials are validated, the Authentication Server will identify the user based on the submitted credentials. SAS Federation Server can then make requests to the Authentication Server for information about the user, including logins and group membership. If a user is authenticated but cannot be identified in the Authentication Server, that user becomes a member of the PUBLIC group. All users that are identified in the Authentication Server are members of the USERS group.

Only individual user objects and shared logins own outbound accounts. Groups, including USERS and PUBLIC, cannot own accounts, including shared logins. If a user is SYSTEM or ADMINISTRATOR, their personal credentials are used to authenticate. SYSTEM and ADMINISTRATOR accounts are discussed further in "Configuring Server Accounts" on page 24.

It is assumed that the Authentication Server has been installed, configured and populated with user and group information before running SAS Federation Server. For more information, see the *Authentication Server Administrator's Guide*.

Authorization

Overview

Authorization determines what privileges a user or group object contains in order to gain access to resources and associated data sources. Because SAS Federation Server requires access to underlying data sources, authorization can happen at two distinct locations:

- Data Source Authorization
- Federation Server Authorization

Data Source Authorization

An example of data source authorization would be an Oracle database, which provides its own layer of security for its data. Data source authorization cannot be bypassed by SAS Federation Server. For example, if an Oracle administrator denies privileges to a user on table T1, then that user will always be denied access to table T1, no matter what privileges are set in SAS Federation Server. SAS Federation Server authorizations are more restrictive on the underlying data.

SAS Federation Server Authorization

About Authorization

SAS Federation Server authorizations are applied to all administration DDL. Most administration DDL is performed by an administrator only (defined as a user who has

the ADMINISTER privilege on SAS Federation Server), but some commands, such as CREATE CACHE, have specific privileges which can be assigned to users and groups.

In the case where a user is connected to data sources providing customer data, SAS Federation Server authorizations are applied over the underlying data source. SQL statements submitted to the server are first parsed and then evaluated against the privileges defined in SAS Federation Server. If the action is not permitted from SAS Federation Server, an error is returned to the user, and no SQL is sent to the underlying data source. If the action is permitted, the SQL statement is evaluated, and the FedSQL processor determines what SQL should be sent to the underlying data sources. In summary, if the underlying data source does not permit the SQL action, then an error is returned. Otherwise, the SQL action is performed and results sent back to the user.

For example, an administrator can configure the server so that a particular user cannot access table T1 even if the underlying data source allows it. So SAS Federation Server authorizations can be used to restrict the type of activity that an administrator wants to allow on the server.

SAS Federation Server authorizations are also very powerful when used in conjunction with shared logins. Shared logins allow many users to be mapped to the same single login for an underlying data source. This allows for easy back-end data source user management, since each user of SAS Federation Server does not require an individual login. However, this alone would mean that all users of that shared login would have the same privileges to the accessible data. However, SAS Federation Server authorization can be used to restrict individual access to data, no matter what the shared login is allowed to access in the underlying data source.

As with other system metadata, SAS Federation Server authorization process uses an internal database to store security definitions for users, groups and objects. Privileges can be set on individual users, or on groups, which affect all members of the group. By default, no users (except those defined as system users) are granted any specific privileges on any objects in SAS Federation Server, and the lack of any privilege anywhere results in a DENY from the server's authorization subsystem. The administrator must specifically grant privileges before a user can perform any actions through SAS Federation Server.

Refer to Technical Support, Object Level Security for best practices related to authorization.

Case Sensitivity

When security definitions are stored in SAS Federation Server, they are stored as entered in the GRANT or DENY SQL syntax. Using the following statement as an example:

```
GRANT SELECT ON TABLE "MyTable" TO BOB
```

SAS Federation Server creates a system table object for table "MyTable". When performing privilege comparisons at run time (for example, when the user Bob issues a SELECT statement against table "MyTable"), the server will perform case sensitive or insensitive comparisons, depending on the data source. Case sensitivity from the data source is registered during the CREATE DATA SERVICE DDL statement, and the system automatically determines the correct setting. However, the administrator can specify case sensitivity as well via the CASE_SENSITIVITY option in the CREATE DATA SERVICE DDL. It is highly recommended that the server default to the correct value. Specifying an incorrect value for case sensitivity can result in incorrect privilege determination.

Case sensitivity settings only apply to relations and columns. Data services, catalogs, schemas and DSNs are always treated in a case-insensitive manner. Also, user and group identifiers are treated as case insensitive.

Permissions

Overview

At times the terms permissions and privileges are used interchangeably. To clarify, permissions represent a specific ability while privileges are a combination of applying the permission to a user and an object. For example, a user's privileges include all the permissions that were granted on various objects in SAS Federation Server.

Permission Types

Federation Server permissions are divided into two general types:

- Administration permissions
- SQL permissions

The following table provides a description for each of these permissions.

Table 5.1 *SAS Federation Server Permissions*

Permission	Description
ADMINISTER	Controls the ability to configure the server. This privilege can be set on the server only. Users who are granted ADMINISTER privilege are automatically granted all other privileges.
ALTER TABLE	Controls the ability to add or drop columns in a table or create or drop indexes with the ALTER TABLE statement. The authorization is set on the server, data service, catalog, and schema objects.
CONNECT	Controls the ability of the user to connect, using either the DSN or data service. The authorization is set on the server, data service, and DSN.
CREATE TABLE	Controls the ability to create new tables or views with the CREATE TABLE statement. The authorization is set on the server, data service, catalog, and schema objects.
DELETE	Controls the ability to delete data with the DELETE statement or analogous method call. The authorization is set on the server, data service, catalog, and schema objects.
DROP TABLE	Controls the ability to remove tables with the DROP TABLE statement. The authorization is set on the server, data service, catalog, and schema objects.
INSERT	Controls the ability to add data with the INSERT statement or analogous method call. The authorization is set on the server, data service, catalog, schema, table, and column objects.
REFERENCES	Controls the ability to create a foreign key reference to an existing table. The authorization is set on the server, data service, catalog, schema, table, and column objects.
SELECT	Controls the ability to retrieve data with the SELECT statement. The authorization is set on the server, data service, catalog, schema, table, and column objects.

Permission	Description
TRACE	Controls the ability of the user to enable tracing and create trace files. This privilege can be set on the server only.
UPDATE	Controls the ability to modify data with the UPDATE statement or analogous method call. The authorization is set on the server, data service, catalog, schema, table, and column objects.
CREATE DSN	Controls the ability of the user to create a DSN. The authorization can be set on the server or data service.
ALTER VIEW	Controls the ability to alter a definer's rights or invoker's rights view. The authorization can be set on the server, data service, catalog or schema objects. [1, 2]
CREATE VIEW	Controls the ability to create a definer's rights or invoker's rights view. The authorization can be set on the server, data service, catalog or schema objects. [1]
DROP VIEW	Controls the ability to drop a definer's rights or invoker's rights view. The authorization can be set on the server, data service, catalog or schema objects. [2]
CREATE CACHE	Controls the ability to create, drop and refresh a cache from a definer's rights view. The authorization can be set on the server, data service, catalog or schema objects, and on individual definer’s rights views.
ALTER CACHE	Controls the ability to enable, disable and refresh a cache. The authorization can be set on the server, data service, catalog or schema objects, and on individual definer’s rights views.
CREATE TABLESPACE	Allows a user to create tables in a schema that they do not own to implement a data cache operation. CREATE TABLESPACE applies to data cache operations. The authorization can be set on the server, data service, catalog or schema objects.
ALTER	Controls the ability to alter a view or a table. The authorization can be set on the table or view object. [1, 2]
DROP	Controls the ability to drop a view or a table. The authorization can be set on the table or view object. [2]

[1] Because definer's rights views effectively allow a user to impersonate the schema owner, the creation of definer's rights views requires special consideration. It can be desirable to grant CREATE VIEW privilege to a user, but only for the intention of creating invoker's rights views. Likewise, if ALTER on a view allows switching invoker’s rights views to definer’s rights views, then that privilege is quite powerful as well. Only the schema owner can create definer’s rights views, along with the system user and administrator, who have all privileges. In addition, only the schema owner can issue an ALTER VIEW command to change an invoker’s rights view or a definer’s rights view. Therefore, a grant ALTER VIEW or ALTER on a view only allows a user to change the view name.

[2] If the ALTER or DROP permission is explicitly set on the object, that privilege definition is honored. If the privilege is not set, the following applies: The proper ALTER or DROP privilege is searched on the container object. If the Alter privilege is on a table, ALTER TABLE is searched. If the privilege is on a view, the search target is ALTER VIEW. This applies to the DROP privilege as well.

Administration Permissions

The administration permissions are:

- ADMINISTER
- CONNECT
- CREATE DSN
- CREATE CACHE
- ALTER CACHE
- CREATE TABLESPACE
- TRACE

Administration permissions are always enforced, regardless of the 'Federation Server SQL Authorization Enforcement' setting for any DSN.

SQL Permissions

The SQL privileges are:

- SELECT
- UPDATE
- INSERT
- DELETE
- REFERENCES
- ALTER TABLE
- ALTER VIEW
- DROP TABLE
- DROP VIEW
- CREATE TABLE
- CREATE VIEW

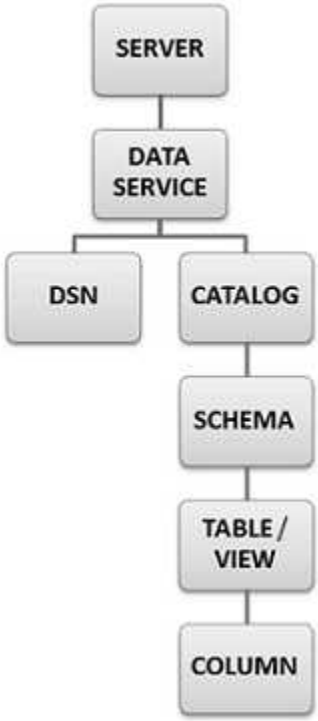
The enforcement of these privileges to users or groups is controlled through the 'Federation Server SQL Authorization Enforcement' option in the DSN. When that setting is disabled, a user connecting with that DSN can perform any SQL action on the data, regardless of what is defined in SAS Federation Server for that user. When the setting is enabled, SAS Federation Server privilege definitions are enforced for that user on that connection. The standard setting for most users will be to enable the SQL authorization enforcement. It will typically only be disabled on DSNs where certain privileged users are granted CONNECT privilege.

Object Privilege Inheritance

About Object Privilege Inheritance

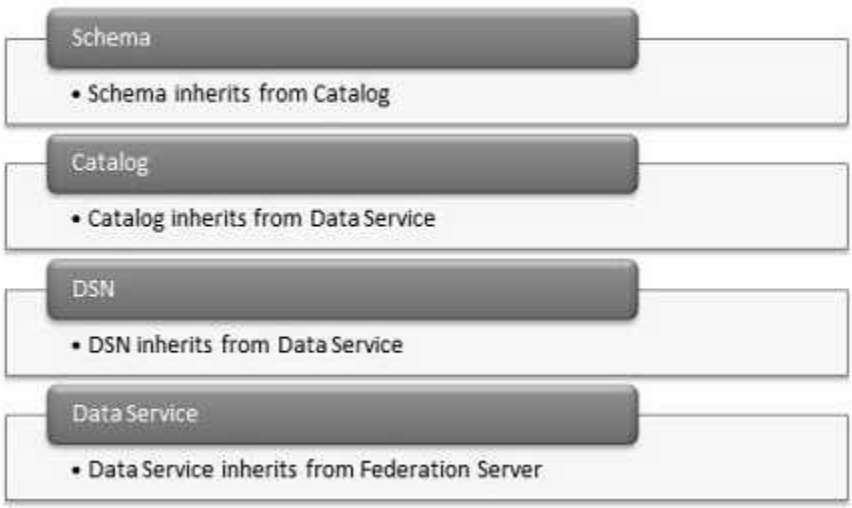
SAS Federation Server contains an inherent hierarchy of objects, in the following order:

Figure 5.1 *SAS Federation Server Privilege Inheritance*



Where privileges on the server are inherited by the data service, privileges on the data service are inherited by the DSN and catalog, and so on. This inheritance hierarchy allows an administrator to set general security rules on higher level objects, and then only set exceptions on the more specific (subordinate) objects.

Figure 5.2 *SAS Federation Server Container Object Inheritance*



For example, there is a group called SALES_GROUP whose members are allowed to select most objects in the SALES_DATA data service. An administrator can assign SELECT privilege on the SALES_DATA data service to the SALES_GROUP. The

SELECT privilege is inherited on all the catalogs of the SALES_DATA data service, and all the schemas of those catalogs, and all the related tables and views. There is a stipulation that the SALES_GROUP is not allowed to see any data in a single catalog of the SALES_DATA data service called EXECUTIVE_DATA. An administrator could then deny SELECT privilege to the SALES_GROUP on that particular catalog. Members of the SALES_GROUP would then not be able to select any data from the EXECUTIVE_DATA catalog or any of its schemas. An administrator can elect to grant all privileges on the EXECUTIVE_DATA catalog to the EXECUTIVE_GROUP. An administrator can also deny SELECT privilege to any member of the EXECUTIVE_GROUP on any subordinate object of the EXECUTIVE_DATA catalog. In this way, general authorizations are defined on higher level objects, and then exceptions are set on subordinate objects. This minimizes the number of privileges that an administrator must establish, and thereby reduces administration overhead. For any object in the hierarchy of SAS Federation Server, an administrator can request information about privileges held by any user or group, including where in the hierarchy the privilege was set and who the grantee of the privilege is, which can be a group that the user is a member of, directly or indirectly.

Object Privilege Summary

The following table summarizes SAS Federation Server objects with their associated privileges. If privileges are inherited, the field is marked as 'Yes'. The fields are blank if there is no privilege inheritance for the object. Inheritance runs from right to left.

Object / Privilege	Column	Row	Table/ View	Schema	Catalog	DSN	Data Service	Federation Server
SELECT	Yes	Yes	Yes	Yes	Yes		Yes	Yes
UPDATE	Yes		Yes	Yes	Yes		Yes	Yes
INSERT	Yes		Yes	Yes	Yes		Yes	Yes
REFERENCES	Yes		Yes	Yes	Yes		Yes	Yes
DELETE	Yes		Yes	Yes	Yes		Yes	Yes
ALTER TABLE/ VIEW TABLE			Yes	Yes	Yes		Yes	Yes
DROP TABLE/ VIEW			Yes	Yes	Yes		Yes	Yes
CREATE TABLE/ VIEW				Yes	Yes		Yes	Yes
CREATE/ ALTERCACHE			Yes	Yes	Yes		Yes	Yes
CREATE TABLESPACE				Yes	Yes		Yes	Yes
ADMINISTER								Yes
TRACE								Yes
CREATE DSN [1]							Yes	Yes

Object / Privilege	Column	Row	Table/ View	Schema	Catalog	DSN	Data Service	Federation Server
CONNECT [2]						Yes	Yes	Yes

Note: [1] Special DSN inheritance applies to CREATE DSN where privileges are checked for inheritance on the data service first and then on the server. [2] For the CONNECT privilege, privileges are first checked on the DSN, and then checked for inheritance on the data service and finally, on the server.

User and Group Privileges

Privileges can be assigned to a user or a group on any object. Group privileges are granted and denied in the same manner as individual users. Users who are members of a group inherit the privileges from the group unless explicitly denied in the individual user account. Also, privileges set on a group that is "closer" to a user take precedence over privileges set on groups that are more distant.

For example, an administrator grants SELECT privilege on table T1 to group G1, and that is the only privilege on that table. If Bob is a direct member of group G1, then Bob has privileges to select from T1. The administrator then denies SELECT privilege on table T1 to group G10, where group G1 is a direct member of group G10. User Bob continues to have privileges to select from T1 because group G1 gives him the SELECT privilege. Group G1 is closer in relation to Bob than group G10—of which Bob is also a member—but indirectly through membership in group G1. If the administrator now denies SELECT privilege on table T1 to group G2, where Bob is also a direct member of group G2, there is a conflict in privileges. Group G1 indicates that Bob is granted SELECT privilege., However, group G2 indicates that Bob is denied SELECT privilege. In these cases where there is a conflict, the user is denied the privilege. In this example, Bob does not have SELECT privilege on table T1.

Maintaining Security Definitions for Tables, Views, or Columns

Overview

If an administrator adds security definitions to a table, view, or column and then later drops the table, view, or column, the security definitions are removed. There might be times when a batch job drops the object and then re-creates it. In this case, the administrator must reestablish security on the object.

Tables

As a suggested best practice, use the DELETE command to remove all of the rows from the table, but leave the table definition intact so that the security definitions remain with the table. Note that indexes created in the data store remain on the table as well. This practice offers an advantage if the table is merely being repopulated.

Views and Columns

There is no way to alter a view definition without dropping the view and re-creating it. In this case, security for the view needs to be reestablished. The same applies for column security. Column security must be reestablished for columns that are dropped and added at a later time.

Granting Privileges

Overview

A system user has no restrictions on SAS Federation Server and can grant any privilege to any user. A SAS Federation Server administrator can grant any privilege on the server to any user except the ADMINISTER privilege itself. In other words, an administrator cannot create other administrators. Only the system user can do this.

Besides system users and administrators, the only other user that can grant a privilege to other users is a DSN owner. This user can grant the CONNECT privilege (and only that privilege) to other users.

Grantor Precedence

The order of precedence between the three groups of users who can grant privileges is as follows:

- system user
- administrator
- DSN owner

Privileges granted from a system user cannot be overridden by administrators, and privileges granted from an administrator cannot be overridden by a DSN owner.

Here are some examples:

- A system user denies SELECT privilege to user John on the server object. An administrator cannot grant SELECT privilege to user John on the server object. An administrator can grant SELECT privilege to user John on a different object, such as a data service, catalog, or schema.
- An administrator grants CONNECT privilege to group SALES_GROUP on DSN SALES_DATA. The DSN owner cannot deny CONNECT privilege to group SALES_GROUP on the same DSN.

Privilege Determination Summary

To summarize, privileges can be determined by group membership and by object hierarchy. The precise algorithm is described here:

- System and administrator users are not denied access to any objects.
- For other users, a specific privilege is first checked at the current object in the hierarchy, in the following order:
  1. On the current object, evidence of the privilege is first searched in the identification of the specific user. If a GRANT or DENY determination is made, the status is returned for the privilege, and privilege lookup stops.
  2. User privilege search continues on the current object under the first level of group membership. If any group indicates DENY, the user is denied access. If all groups indicate GRANT, then the user is granted access. If a privilege determination cannot be made, the next level of group membership is checked.
  3. On the current object, the privilege is searched for under the next level of group membership, per the same rules as the previous item. If no specific determination is made, repeat for all levels of group membership.

4. If no determination is made on the current object, then privilege determination goes to the next higher level in the object hierarchy. The privilege search algorithm repeats as described, first under the identification of the specific user, and then group membership.
5. If all objects in the hierarchy are searched and no privilege determination is made, then a DENY is returned for the privilege type for the use.

Privilege Caching

Overview

Privilege caching entails capturing and reusing privileges enforced on a statement or request submitted to SAS Federation Server. This improves performance by reducing queries against internal system tables.

Privileges are cached on demand. Each time privileges on a securable for a given grantee are checked, the cache is examined initially to see whether the privilege has already been cached. If so, enforcement cost is reduced by limiting or eliminating queries against security-related system tables. Once uncached privileges are retrieved from system table queries, they are cached for subsequent use, stabilizing the cache based on actual access patterns.

Configuring Privilege Caching

Use the ALTER SERVER DDL statement on page 159 to cache privileges. For example:

```
alter server {options xset CACHE(name 'Authorization',
level <x level>, purge )}
```

- The default level is 2.
- The purge option frees the current cache. If privilege caching is enabled, the system dynamically builds the cache again.
- Use the purge option to trim memory usage in a long-running server instance.

Level Control

0	Privilege caching is disabled. All levels are purged.
1	Schema, catalog, server, and service-level privileges are cached. Level 2 is purged.
2	Column, row, table, schema, catalog, server, and service-level privileges are cached.

Row-Level Security

Introduction

Row-level security (RLS) for SAS Federation Server provides additional granularity of security on tables and views. It allows only certain rows to be selected for a given set of users and groups. Because row-level security only applies to rows that can be selected, it

is implemented as part of the SELECT privilege. The SELECT privilege can be granted without restriction, or with a predicate applied. When a predicate is applied, this is called row-level security because the predicate will restrict the rows returned.

For example, an administrator might choose to grant SELECT privilege to USER1 on table T1. In this case, USER1 is allowed to see all the data in the table. But an administrator might allow USER1 to only select from rows where a column or set of columns meet certain criteria, such as where the sales region is the northeast. In this case, the grant statement looks like:

```
GRANT SELECT ON TABLE CATALOG.SCHEMA.T1 TO SALES
WHERE SALES_REGION = 'NORTHEAST'
```

In this case, when members of the group SALES select from table T1, the predicate is automatically attached to the table when BOB, a member of the SALES, issues the statement:

```
SELECT * FROM T1
```

His query is effectively transformed to:

```
SELECT * FROM T1 WHERE SALES_REGION = 'NORTHEAST'
```

Reference Administration DDL on page 159 for syntax details about the GRANT statement.

Because the predicate is automatically attached to the table, it must contain a valid WHERE clause. The syntax can include sub-queries and references to other tables. However, any external data referenced in the predicate must be available on the user's connection and the user must have the SELECT privilege to access the data.

If the data is coming from a different data source, then:

- The user's DSN must scope to that data source.
- The user must have SELECT privilege on the referenced data.

For example, a user can select rows only from a table for which the user's name is listed in the "USER NAME" column of the table. To apply this rule to all members of the SALES group, an administrator can issue the following GRANT statement:

```
GRANT SELECT ON TABLE CATALOG.SCHEMA.T1 TO SALES WHERE
SYSCAT.RLS.CURRENT_USER() = \"USER NAME\"
```

When the user BOB in group SALES selects from the table:

```
SELECT * FROM T1
```

The query is transformed to:

```
SELECT * FROM T1 WHERE 'BOB' = "USER NAME"
```

The RLS Library and Library Reference on page 80 contains details about callable functions for row-level security.

Row-Level Security Privilege Assembly Rules

Overview

The SELECT privilege for a user can be derived through group memberships. A user can be a member of multiple groups, each granting SELECT privilege with attached row-level security. One exception is the schema owner. Since a schema owner effectively has all privileges granted, group membership is not traversed for privileges at the schema level.

RLS predicates are assimilated at each level in the securable hierarchy, starting with the table or view object and progressing to the server object.

The procedure is repeated over each object in the authorization hierarchy:

- the current user. If the current user has an RLS condition applied, no other RLS conditions are considered.
- any groups of which the current user is a member.
- followed by USERS and PUBLIC only if no other RLS conditions were discovered.

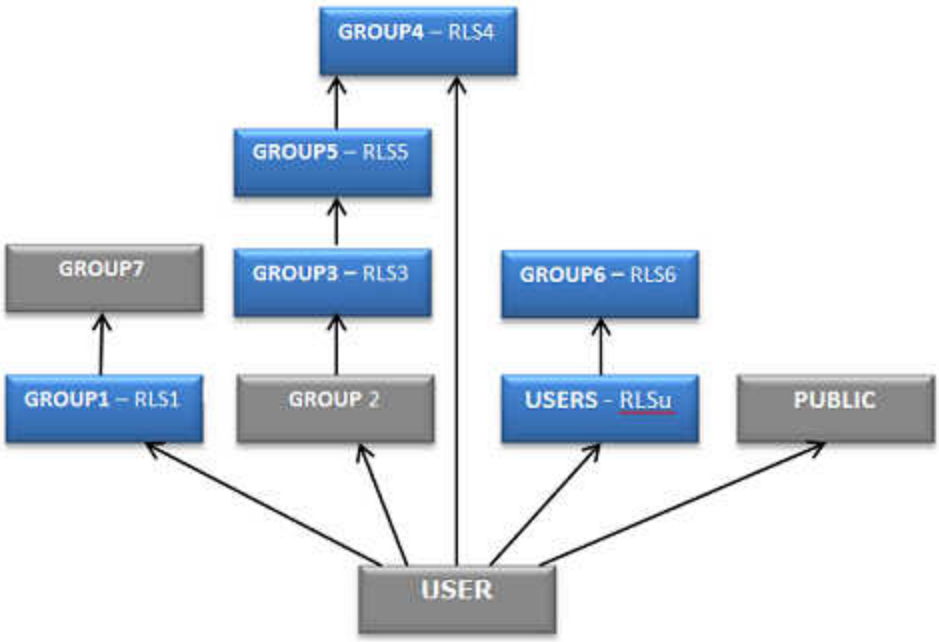
If an RLS predicate is discovered at a specific level, it is combined with the other RLS predicates at that level only, using an OR operator, and the process stops. The procedure will produce predicates representing the summation of RLS privileges closest to the user. This approach gives preference to organizational security policies closer to the user over those more distant. An unconditional GRANT or DENY at the same level as an RLS predicate will be honored, and all RLS predicates will be ignored.

Example: RLS Predicate Union

In this example USER is a direct member of the user-defined groups GROUP1, GROUP2 and GROUP4. USER is also a member of the two built in groups, USERS and PUBLIC. The RLS query returned for USER is (RLS1 OR RLS4 OR RLS3).

Note: Blue nodes contain a conditional privilege; gray nodes contain NO privilege.

Figure 5.3 *RLS Predicate Union*



The first predicate, RLS1, is encountered at level 1 in the graph, so the remaining RLS predicates are captured at that level for the current graph, which does not include the secondary graphs, USERS and PUBLIC. The GROUP1 node is marked as visited, and its parent ("member of") associations will not be navigated since it contributed a predicate.

After marking it as visited the procedure skips GROUP2 since it has no assigned privileges and proceeds to GROUP4 where the node is marked as visited and RLS4 is

captured and combined to the query using an OR operator. There are no more nodes at level 1, so the procedure continues with the parent associations of GROUP2.

GROUP3 is marked as visited and predicate RLS3 is captured and combined to the query using an OR operator. The privilege has been satisfied at this point because an RLS query is available. The two graphs starting at USERS and PUBLIC are not searched.

The resulting RLS query is: (RLS1 OR RLS4 OR RLS3).

The RLS Library and Library Reference

Overview

Use the RLS library to look up functions that reference your data source. The RLS library resides in the Federation Server Database as SYSCAT.RLS. The general syntax is **`SELECT syscat.rls.RLS_function('name').`** For example:

```
SELECT syscat.rls.group_id('GROUP1')
```

RLS Library

In addition to row-level security, RLS Library user functions can be used with other SAS Federation Server tasks such as FedSQL views and queries by including an RLS function in your SELECT statement: **`syscat.rls.rls_function`**.

The following table lists the callable functions for row-level security:

Function	Description
auth_id	Returns the authentication identifier.
current_user	Returns the name of the current user.
current_id	Returns an opaque authentication provider specific user identifier.
domain	The name of the domain in which the current user is authenticated.
group_id	Returns the group identifier
login	The domain-qualified login used to authenticate the current user.
userid	Similar to login. However, userid is not domain-qualified.
ip_addr	Returns the client IP address of the current user’s session.
is_admin	Returns TRUE (FALSE) if the current user is (not) an administrator.
is_process_user	Returns TRUE (FALSE) if the current user is (not) the process user.

Function	Description
member_of	Returns TRUE (FALSE) if the current user is (not) a member of the specified group
groups	Returns a single group name or identifier column result set containing the current user's group memberships.

Library Reference

WVARCHAR(n) auth_id(WVARCHAR(n) [[authorization]])

Returns an authentication identifier as defined by the authentication provider, as a result of passing input for user name.

WVARCHAR(n) current_user()

Returns the name of the current user. This is the FedSQL authorization identifier of the currently authenticated user, rather than the login used.

WVARCHAR(n) current_id()

Returns a user identifier as defined by the authentication provider. Typically, this is a static identifier by which the current user is known. Applications can associate this identifier with an internal organization user identifier such as an employee number or account number.

WVARCHAR(n) domain()

The name of the domain in which the current user is authenticated.

WVARCHAR(n) group_id(WVARCHAR(n) [[authorization])

Returns a group identifier as defined by the authentication provider, as a result of passing input for group name.

WVARCHAR(n) login()

The login used to authenticate the current user.

WVARCHAR(n) userid(BITupn)

The domain qualified user ID. If the upn parameter is TRUE, the format of the returned user ID is user@domain. Otherwise, the format is domain\user on Windows systems and just userid on all other systems. The userid function returns the authenticated user ID as specified by the authentication service. The authentication service can reside on a different host.

WVARCHAR(n) ip_addr()

Returns the client IP address of the current user's session.

BIT is_admin()

Returns TRUE or FALSE if the current user is or is not an administrator.

BIT is_process_user()
Returns TRUE or FALSE if the current user is, or is not the process user.
BIT member_of(WVARCHAR(n) group [, WVARCHAR(n) options])
Returns TRUE or FALSE if the current user is, or is not a member of the specified group. Can assert direct or indirect membership. The group parameter is a group name by default and a group identifier if the ‘ID’ or 'DEEP' option is present in the options string. The options string is a blank or comma separated string consisting of one or more of ‘ID’ and ‘DEEP’ option keywords. The current user must be a direct member of the specified group unless the ‘DEEP’ keyword is specified. Direct membership is tested by default.
TABLE(WVARCHAR(n) group) groups(WVARCHAR(n) [[authorization] WVARCHAR(n) [, options]]
Returns a single group name or identifier column result set containing the current user’s group memberships. The available options are ‘ID’ or ‘DEEP’. Can be restricted to direct memberships only. The authorization parameter is a user or group name by default and a user or group identifier if the 'ID' option is present in the options string. The options string is a blank or comma separated string consisting of one or more of ‘ID’ and ‘DEEP’ option keywords. A deep group membership listing is returned if the ‘DEEP’ keyword is specified, the default being a shallow listing.Note: A trusted user must be set in order for the GROUPS function to return a result set if you pass in a user other than the current user. If you pass in the current user or a group as the first argument to GROUPS trusted user is not required.

Usage Notes for the member_of and groups functions returning ‘ID’ or ‘DEEP’ member_of and groups functions:

```
SYSCAT.RLS.GROUPS(user_name_or_id [, options])
```

Lists the groups that user_name_or_id is a member of.

```
SYSCAT.RLS.MEMBER_OF(group_name_or_id [, options])
```

Returns TRUE if the current user is a member of the specified group_name_or_id group.

- **user_name_or_id**: a string literal indicating the user. If **`options`** contains ‘ID’ this will be treated as a user ID (for example, '6C6C9AD1E2646F0469DD6A3D1874D167'). Otherwise, this will be treated as a user name (for example, 'USER1').
- **group_name_or_id**: a string literal indicating the group. If **`options`** contains ‘ID’ this will be treated as a user ID (ex: '78319AD1E2646F0469DD6A3D1874A2F7'), otherwise this will be treated as a user name (ex: 'GROUP1').
- **options**: a string literal containing 'GROUP', 'ID', or both (ex: 'GROUP, ID'). If multiple options are specified, they can be separated in the string by a blank ' ' or comma ','. If **`options`** contains 'ID' the first argument is treated as an ID rather than a name. If **`options`** contains ‘DEEP’ group membership will be checked recursively.

For example, consider that USER1 is a member of GROUP1 and GROUP1 is a member of GROUP2. USER1 runs the following queries:

```
Select SYSCAT.RLS.GROUP_ID('GROUP1') returns

'BEA892C6D4A40464C8A144D89FFE6463'

Select SYSCAT.RLS.GROUP_ID('GROUP2') returns

'45C562900C7333C49B1706B38DBA75B0'

Select SYSCAT.RLS.CURRENT_ID() returns

'5790EE3F6A24A7349AA2254600793411' for USER1
```

```
Select SYSCAT.RLS.MEMBER_OF('GROUP1') returns

 TRUE for USER1

Select SYSCAT.RLS.MEMBER_OF('GROUP2') returns

 FALSE for USER1

Select SYSCAT.RLS.MEMBER_OF('GROUP2', 'DEEP') returns

 TRUE for USER1

Select SYSCAT.RLS.MEMBER_OF('BEA892C6D4A40464C8A144D89FFE6463', 'ID') returns

 FALSE for USER1

Select SYSCAT.RLS.MEMBER_OF('45C562900C7333C49B1706B38DBA75B0', 'ID') returns

 TRUE for USER1

Select SYSCAT.RLS.MEMBER_OF('BEA892C6D4A40464C8A144D89FFE6463', 'DEEP, ID')
returns

 TRUE for USER1
```

The following queries against GROUPS return a results set:

```
Select * from SYSCAT.RLS.GROUPS('USER1') returns

 "GROUP"

 `USERS`

 `GROUP1`

`PUBLIC`


Select * from SYSCAT.RLS.GROUPS('USER1', 'DEEP') returns
   "GROUP"
   'USERS`
   `GROUP1`
   ‘GROUP2’
   `PUBLIC'
```

Server Encryption

Introduction

SAS Federation Server supports two methods of encryption strength: SAS Proprietary and DataFlux Secure.

SAS Proprietary

SASProprietary is a fixed encoding algorithm that is included with SAS Federation Server. It requires no additional product licenses and is the default encryption method if DataFlux Secure is not installed. The SAS proprietary algorithm is strong enough to protect your data from casual viewing. SASProprietary provides a medium level of security. SAS/SECURE and SSL provide a high level of security.

DataFlux Secure

Overview

DataFlux Secure is an add-on product that provides industry encryption capabilities in addition to the SASProprietary algorithm. DataFlux Secure requires additional licensing and it must be installed on each server that will use encryption. DataFlux Secure provides encryption of data in transit. It does not provide authentication or authorization capabilities.

The AES – 256-bit keys encryption algorithm is used by SASProprietary and DataFlux Secure.

Specifying the Encryption Method

By default, the encryption is set to SASProprietary. You can also decide how much data is encrypted in communication between a client and SAS Federation Server. This is specified by setting the CLIENTENCRYPTIONLEVEL using the **ObjectServerParms** option in **`dfs_serv.xml`**. See the SAS Federation Server Configuration Reference on page 39 for more information on this option.

Password Encryption

SAS Federation Server provides a utility to encrypt user passwords from plain-text format. The encryption method depends on the encryption method in use for SAS Federation Server. See “Utilities for SAS Federation Server ” on page 63 for additional information about password encryption.

Chapter 6

Working with Federated Data

Overview of Data Federation

By supporting several data sources, SAS Federation Server provides the flexibility to configure data storage based on specific needs. You choose the type of data storage that

is most appropriate for the particular needs of an application, based on functionality that is provided by each data source. The first step in working with federated data is to configure access to your data sources.

Before creating federated (FedSQL) data views and caching data, make sure that your trusted user and shared logins are configured and ready for use. and have been configured and are ready for use. For additional information, see"Configuring Server Accounts" on page 24.

For a comprehensive view of data federation and best practices, refer to "Understanding Data Federation and Best Practices" on page 104 at the end of this chapter.

Configuring Data Source Access

Overview

To access data, the administrator must create and configure data services for SAS Federation Server. Data services contain connection information and driver specifics to connect with data sources such as Oracle® or Base SAS® data sets.

Relational databases provide authorization that limits the operations that can be performed on their data. As noted previously in "Data Source Authorization" on page 68, SAS Federation Server respects authorizations that are defined and enforced on a third-party database.

Data Services

Overview

Data services contain information that identifies the location of data tables. If a data source does not support native catalogs, SAS Federation Server enables you to define a logical catalog name to use as an SQL identifier. This allows each data source to be uniquely identified when performing heterogeneous operations.

Data services requiring logins must be associated with a domain in the Authentication Server. When users connect to the data service through a data source name, the domain name is used to retrieve user credentials associated with that data service. The credentials are then passed along to the back-end database. User credentials are stored in the Authentication Server.

Data services can also contain optional information to control SAS Federation Server driver behavior, such as locking semantics and tracing. Data services form the foundation for connectivity to a source of data, and privileges can be assigned to data services to control user access to the given data service.

Creating a Data Service

Use administration DDL or SAS Federation Server Manager to create a data service.

By default, a BASE data service is created the first time that SAS Federation Server is started. Only one BASE data service can exist in a SAS Federation Server installation, and it cannot be modified or deleted.

Native Catalog Support

When creating a data service for a data source that supports native catalogs and using the REGISTER option, the server attempts to connect to the database to acquire a list of catalogs. Credentials are required to secure the connection. If the connection cannot be made, creation of the data service will fail. The same requirement for pre-registered credentials applies when creating a data service to ODBC with native catalog support, or for any data services that support native catalogs.

Identifier Case Sensitivity

When creating an ODBC data service, the server must query the data source to acquire its identifier case sensitivity property. The identifier case sensitivity property is used to create security entries in the Federation Server system tables and is stored with the data service.

Database Login Prerequisite

Due to the requirement for a database connection described above, the following database login prerequisite applies to SQLSERVER, ODBC and ODBC_FED data services. These actions must be completed before creating the data service, and must be accomplished through the Authentication Server Manager.

1. A database login must be registered in the domain that will be associated with the new data service.
2. The domain must be registered in Authentication Server for it to be accessible when creating the data service using administration DDL or SAS Federation Server Manager.

Creating a Data Service with Administration DDL

Here is an example of the DDL statement, CREATE DATA SERVICE. See CREATE DATA SERVICE DDL on page 163 for details and a complete list of options. You can also alter and drop data services using DDL.

```
CREATE [DATA] SERVICE data-service
TYPE data-service-type
[CATALOG [NAME] catalog-name]
[DOMAIN [NAME] domain-name]
[REGISTER [( catalog-name1 [,catalog-name2 …]) | ALL]
[register-options]]
[data-service-options]
```

Registering Catalogs

Use CATALOG to register a catalog name for data sources that do not support native catalogs.

Use REGISTER to register the native catalog names for those data sources that support native catalogs, such as SQL Server. If an identical catalog name is encountered, a warning message is issued and the catalog is not registered. In this case use the CREATE CATALOG DDL on page 168 statement to provide a mapped name for the native name.

Driver Search Order

If specifying more than one driver in the CREATE DATA SERVICE statement, the first driver listed in the statement is the default driver used to connect to a data source if a driver is not specified in a connection string. For example, if two drivers are specified when creating a data service, the first driver is used as the default driver if **driver=** is

not specified in a connection string. If a driver is specified in a connection string using conopts-configuration-list when creating the data service, it is used when connecting to a data source.

Catalogs and Schemas

Overview

The terms catalog and schema are defined as ANSI SQL standards and refer to the organization of data in a relational database. That is, data is contained in tables, tables are grouped into schemas, and schemas are grouped into catalogs. Catalog and schema names can be used in SQL statements to qualify table references. For example, when querying a database that supports both schemas and catalogs, you can specify a three-level identifier in the form of CATALOG.SCHEMA.TABLE-NAME.

Organize Data with a Catalog

A catalog is a named collection of logically related schemas. The catalog is the first-level (top) grouping mechanism in a data organization hierarchy that qualifies schemas. At least one schema is required for each catalog.

For the BASE data service, you must create catalogs and schemas in order to make data available. For all other data services, catalogs and schemas are defined in the data source (though the data source is not required to support either catalogs or schemas), and catalog and schema names can be registered in SAS Federation Server to reflect those objects.

Catalog Registration

Catalog names for all data sources must be registered in SAS Federation Server and they must be unique within the system.

This is accomplished by using one of the following methods:

- Use the CATALOG keyword on the CREATE DATA SERVICE command. Do this when the data source does not support native catalogs.
- Use the REGISTER keyword on the CREATE DATA SERVICE command. Do this when the data source supports native catalogs.
- Use the CREATE CATALOG command. Do this to provide a mapped name for a native catalog that cannot be registered using the REGISTER keyword because it conflicts with an existing registered catalog.

The following is a sample of the CREATE CATALOG DDL statement:

```
CREATE CATALOG catalog UNDER data-service
[ NATIVE NAME native-name ]
[ create-catalog-options ]
```

A complete list of options is shown in the CREATE CATALOG DDL statement on page 168 .

Organize Data with a Schema

A schema is a data container object that groups logically related objects such as tables and views. The schema provides a unique namespace that is used along with a catalog to qualify names.

For SAS data sets, a schema identifies the physical location such as a UNIX® directory or a Windows® folder that contains a collection of tables. That is, for SAS data, the

relationship between a schema and its files is similar to that of an operating system file directory and the files that are contained within that directory. For SAS data, a schema is approximately equivalent to a SAS library.

Schema Registration

Unlike catalogs, schema registration is not required for all schemas in the data source. Schemas are registered only when the administrator wants to assign an owner to the schema. Schemas are also created and maintained internally as needed by the system, such as when assigning privileges to a user or group on a schema.

The following is an example of the CREATE SCHEMA DDL statement:

```
CREATE SCHEMA [ catalog.schema ]
[ AUTHORIZATION|OWNER owner ]
[ create-schema-options ]
```

A complete list of options is shown in CREATE SCHEMA DDL on page 171 .

Schema Ownership

All schemas have an owner. If an owner is not explicitly assigned to a schema, ownership defaults to the system user account. Definer's rights views require a non-system schema owner for proper operation. The schema owner is the owner of all objects contained in the schema, though the owner has particular relevance to definer's rights views. And as the owner, certain privileges are automatically granted to the schema owner.

Privilege Rules for Schema Ownership

Here are additional rules that apply to schema ownership:

- The schema owner automatically has all SQL privileges on tables and views in the schema. They are reported with GRANTOR=ADMINISTRATOR. Also, the schema owner can alter the schema's configuration options.
- Administrators can DENY a privilege on the schema, and the owner will be denied the privilege. This feature can be used to downgrade schema ownership rights. Therefore, the schema owner has no explicit privileges on the schema, but has default GRANT for privileges on schema objects.
- Administrators can reverse denied privileges using GRANT. Reference #4 below.
- GRANT to schema owner is equivalent to a REVOKE. The command clears any explicit denied privileges on the schema, but does not add any explicit ones. That way, when a schema owner's privilege is cleared on the schema, it defaults back to implicit GRANT.
- When schema ownership changes, the previous owner receives default privileges from the schema's container, whether it be a catalog or a data service, but retains any explicitly denied privileges on that schema.
- Schemas should not be owned by any user who is a system user.

Configuring DSNs

Overview

DSNs are used to expose data services to users connecting through SAS Federation Server. Typically, when a client connects to SAS Federation Server, they will specify a DSN. The administrator can assign privileges to determine which users can connect. In

order to connect to a data source, a user must be granted CONNECT privilege on SAS Federation Server, a specific data service, or a specific DSN.

A DSN references a specific data service to which it will connect and defines how SQL security is enforced. It can be configured so that SAS Federation Server enforces SQL privileges defined for the data service. The CREATE DSN privilege is required to create a DSN. Configure DSNs using DDL statements or by using SAS Federation Server Manager. For an explanation of CREATE DSN DDL syntax, refer to Administration DDL, CREATE DSN statement on page 174.

DSN Types

Standard DSN

A standard, single-service DSN is created for one particular data service and is parented to that data service. The scope is limited to one data service and contains connection information, such as server name, port, path or other connection options specific to a data service.

Auto-Created DSNs

A new DSN is automatically generated each time a new data service is created. This is a standard DSN with the same name as the data service. The DSN name remains consistent even if the data service is renamed.

SQL_LOG DSN

When SQL Logging is enabled on SAS Federation Server, a SQL_LOG DSN and data service is created automatically. Also, the server creates an EVENTS table in the SQL_LOG data service. This table contains data captured around certain activity in the server, such as information about SQL statements submitted by connected users. Additional information can be found in “SQL Logging” on page 49.

Federated DSN

A federated DSN is a collection of one or more standard DSNs. Unlike the standard DSN which is parented to a data service, the federated DSN is parented to the federation server itself, even if it only contains DSNs from a single data service. Federated DSNs can contain other federated DSNs. Since federated DSNs are typically used to provide access to data from multiple, disparate data sources, the FedSQL dialect is required.

The ADMIN DSN

The ADMIN DSN is created automatically at server startup and is used for the purpose of sending administration SQL to the server. The ADMIN DSN is also used to query Information Views on page 188. The SAS Federation Server Manager automatically connects with the ADMIN DSN to display information such as the registered list of data services and DSNs. Any user expected to use SAS Federation Server Manager to accomplish tasks, such as creating views and caching views, will require CONNECT privilege to the ADMIN DSN. SAS Federation Server automatically checks a user’s privileges when any administration SQL is submitted. Some administration SQL can be submitted by any user (for example, selecting against Information Views for which they have privileges), while other administration SQL can be executed by the server administrator only. See the *SAS FedSQL Reference Guide* for details.

CONNECT privileges are not assigned to DSNs by default, and must be granted by the administrator or by the DSN owner, to users or groups that are connecting to data sources. See the GRANT and DENY statements on page 183 for additional information.

Common Configuration Options

Enabling Federation Server SQL Authorization Enforcement

When Federation Server SQL Authorization Enforcement is enabled, the FedSQL driver is also required, and the SQL dialect is automatically set to FedSQL. With FedSQL an additional layer of object-level security is enabled for the connection and SQL statements are secured before processing them. If Federation Server SQL Authorization Enforcement is disabled, object-level security is bypassed and a user is granted all privileges regardless of what the user has been granted or denied. If Federation Server SQL Authorization Enforcement is disabled, an administrator can choose either FedSQL dialect or data source (native) dialect. For example, if you are connected to Oracle, then native dialect would be SQL supported by Oracle. The SQL dialect for Base data services is always FedSQL.

Security is enabled by default for all new DSNs. However, if you need to enable SAS Federation Server security on a DSN, use DDL options with CREATE DSN and set SECURITY to YES.

Here is an example DDL statement that enables SAS Federation Server security:

```
CREATE DSN "DSN1" UNDER BASE
DESCRIPTION 'creating DSN1' NOPROMPT
'DRIVER=BASE;CATALOG="catalog1_BASE";SCHEMA=(name="schema1_BASE")' {OPTIONS
(SECURITY YES)}
```

Login Credentials

If data services require credentials, the DSN can be configured to specify how database logins are retrieved. The DSN can be configured to use the personal credentials of the user, or retrieve the login from a shared login. If you are using a shared login, you can specify a consumer group from the DSN. This is required only to identify what shared login to use if multiple shared logins are available in the same domain to connecting users.

When using SAS Federation Server Manager, an administrator can have either personal credentials or a shared login to the underlying databases for the purpose of managing data services. SAS Federation Server Manager connects to a data service behind the scenes and data services use a credential search order of PERSONAL, SHARED (CSO=PERSONAL,SHARED).

Note that this means:

- If an administrator has both a personal and a shared login, the personal login will be chosen.
- If an administrator does not have a personal login, but has multiple shared logins available, the connection may be disallowed. See “Best Practices for Setting Shared Logins” on page 28 for rules on how shared logins are selected.

Credentials Search Order (CSO) for DSN Connections

Connections made using a DSN use a credentials search order (CSO) as specified in the DSN configuration. By default, the credentials search order is PERSONAL, SHARED. Other valid values are SHARED, (PERSONAL, SHARED) and (SHARED, PERSONAL).

The server will attempt to select a user ID and password for each data service connection request based on the domain associated with it:

- PERSONAL means the server will attempt to select credentials directly owned by the user.
- SHARED means the server will attempt to select credentials from a shared login of which the user is a consumer. Credentials are extracted on behalf of the user using the shared login manager's identity. Shared Logins must be configured in SAS Federation Server and defined in Authentication Server for this option to function properly. See Shared Login Manager Configuration on page 28 for more information.
    - If a DSN is configured as CSO(SHARED) and a shared login is not found for any of the DSN's connections, the connection will fail immediately.
    - If the credentials search order is not configured on the DSN or is not CSO(SHARED), the connection will still be attempted. If credentials are specified on the connection string, those will be used first. If credentials are not supplied, the server will attempt to find shared logins for the user. If shared login credentials are not found, the server will attempt to use personal credentials. If no personal credentials are found, the connection will fail.

Also, if GROUP=groupname is specified as part of the DSN used to connect or supplied directly in the connection string, shared login selection is limited to those candidates in which the specified group is a consumer. The user must be a direct or indirect member of the group, or no shared login will be selected at all. The GROUP= option does not have any effect on personal login extraction.

DSN Permissions

DSN permissions are assigned using GRANT, REVOKE, or DENY DDL statements.

The permissions for standard and federated DSNs are:

- CREATE DSN
- CONNECT

See the topic on Permissions on page 70 for additional information about permission assignment.

CREATE DSN
To create a DSN, one of the following must be true:

- User is a System User
- User is the Administrator of the server
- User has CREATE DSN privilege on the server (this is the only way a non-admin can create a federated DSN)
- User has CREATE DSN privilege on the data service, and the user is creating a standard DSN

ALTER/DROP DSN
In order to alter or drop a DSN, one of the following must be true:

- User is a System User
- User is the Administrator of the server
- User is the owner of the DSN

CONNECT Permission
A user must have CONNECT permission to establish a connection to a DSN. This permission is effective from the user object, inherited through the hierarchy, or obtained though group permissions.

For a standard DSN, the CONNECT privilege must be on (in order of inheritance):

- The DSN,
- The parent data service of the DSN, or
- The SAS Federation Server

For a federated DSN, the CONNECT privilege must be on (in order of inheritance):

- The DSN
- The SAS Federation Server

Note: If a user is given CONNECT permission on a federated DSN, privileges on any contained DSN (standard or federated) are ignored. If a user has CONNECT permission on a federated DSN, it does not matter if the user is explicitly denied CONNECT privilege on any child DSN.

Creating DSNs using DDL Statements

Using administration DDL, you can create standard and federated DSNs with various configuration options. For a complete list of configuration options, refer to the CREATE DSN DDL statement on page 174 .

Standard DSN
Here is the DDL syntax for creating a standard DSN under a data service:

```
CREATE DSN dsn-name UNDER data-service
          create-dsn-options [ AS ADMINISTRATOR ]
```

Note: If a DSN is created by a user other than the system user or administrator, the DSN is owned by the individual user. If that user is later removed from the system, DSN ownership should be transferred to another user.

Federated DSN
Federated DSNs are objects of SAS Federation Server. Therefore, they are not created under a data service. Following is the DDL syntax for creating a federated DSN:

```
CREATE DSN dsn-name
            create-dsn-options
           ADD "(" dsn-name ["," ...] ")"
```

Federated Data (FedSQL) Views

Overview

When there is a need to view information from multiple data sources or other source types, you can create a reusable FedSQL view to deliver data from multiple relational and non-relational sources. A federated data view contains the information required to access database sources and can be stored separately from the data. By creating a view

definition, you are storing only the instructions for where to find the data and how it is formatted, not the actual data.

Views can reduce the impact of data design changes on users. For example, you can redirect data sources or change variables that are stored in a view without changing the characteristics of the view's result. The view remains consistent even if the data source changes. To create FedSQL views, the FedSQL dialect must be selected in the DSN.

FedSQL Views as Data Abstraction

The concept of data abstraction is used in database systems to define user interfaces through the creation of database views. Based on the data abstraction layer concept, a FedSQL view hides the complexity of data by defining an organized data structure for presentation to an end user or calling application. The result is that a user or application can request data in the organized virtual format, without regard to the physical layout. Data is fetched from potentially many data sources, transformed into the virtual structure and returned to the user or calling application.

Invoker and Definer's Rights Views

There are two types of FedSQL views for SAS Federation Server:

- **Invoker's rights**: An invoker's rights view is run with the invoking user's credentials.
- **Definer's rights**: A definer's rights view is run with the credentials of the schema owner.

The invoker view is accessed using the current user’s authorization, credentials, and login information while the definer's rights view is accessed using the schema owner’s authorization, credentials and login information. A definer's rights view is always associated with a schema owner.

Definer’s rights views allow security to be managed more simply from a single layer of data, which in turn provides for a more secured system. For example, there are 100 tables that provide data to a set of users. 10 views are created and their data is acquired from the 100 tables. The users are selecting from the 10 views to get their data. With invoker's rights views, each invoker must have access to the 100 tables. This includes setting privileges in SAS Federation Server, and ensuring that each invoker has a login to the data sources containing the 100 tables.

With definer's rights views, the data available through the view is accessed by a single user only: the schema owner. Therefore, only this user, the schema owner, needs server privileges and database logins to the data sources containing the 100 tables. View invokers do not need direct access to the underlying tables. The administrator can secure the definer's rights view using Federation Server SQL authorizations to control which users and groups have access to the view’s result set. Unless explicitly specified as definer views, views are created as invoker views.

Here is an example of creating an Invoker view:

```
CREATE VIEW
            view1 AS SELECT * FROM table1
```

Definer's rights views are required for data caching. Only a schema owner can create a definer’s rights view for a schema that he owns. A non-schema owner cannot create a definer’s rights view for a schema that he is allowed to select from. Create a definer view using the following example syntax:

```
CREATE VIEW
```

```
view1 SECURITY DEFINER AS SELECT * FROM table1
```

The following example alters an existing view to invoker's rights or definer's rights. If a definer view has any associated cache, it will be dropped when the view is changed to Invoker.

```
ALTER VIEW
            view1 SECURITY INVOKER

ALTER VIEW
            view1 SECURITY DEFINER
```

Requirements for Definer's Rights Views

Note: This section uses the VSO acronym to indicate View Schema Owner, which is the owner of the schema where the view resides.

Here are the requirements for definer's rights views.

- Use of definer's rights views requires a trust relationship between the Federation Server and the designated Authentication Server. Trust is established in the Authentication Server when connected using a trusted account. This enables impersonation of schema owners during SQL execution. Specifically, the capability is required to retrieve (1) group memberships required for SQL authorization enforcement of data accessed in definer's rights view execution and (2) outbound database credentials forwarded to make transient connections used during view execution.

  Define a Trusted User in the Authentication Server's configuration file. See *Authentication Server Administrator's Guide* for additional information.

  Specify the Trusted User in the Federation Server by setting the trusted user account and password via an ALTER SERVER command:

  ```
  ALTER SERVER {OPTIONS ( TRUSTED_USER_UID uid, TRUSTED_USER_PWD pwd )}
  ```

- Register all database catalog names referenced from the definer's rights view.

  In order for a definer's rights view to access data from a SQL Server data service (or any data service that supports native catalogs), the Federation Server administrator must have pre-registered the referenced catalog names.

  To ease administrative burden, the Federation Server administrator can automatically register catalogs when creating SQL Server data services via the REGISTER keyword of the CREATE DATA SERVICE command:

  ```
  CREATE DATA SERVICE data-service TYPE data-service-type REGISTER
  ```

- Any user that is allowed to create, alter, or drop a view within a schema should be granted CREATE VIEW (ALTER VIEW, DROP VIEW) privilege on the schema or an object in its inheritance hierarchy. The schema owner implicitly has these privileges.
- A schema owner must be assigned to the schema that the view resides in. This is the view schema owner, also referred to as the VSO. This user owns all definer's rights views within that schema, and the VSO user is used for executing the uncached view. Authorization enforcement for all SQL data accessed by the view query is performed using the identity of the owner rather than the invoker.
- The VSO must have CONNECT privilege for the data service where the view is located.

- The VSO must own a database login to the database in which he is the schema owner, assuming that the database requires a login.
- The VSO must own database logins necessary to connect to the database accessed by the view query.
- The VSO must have CONNECT privilege on all data services referenced in the view (data service references are based on the catalog names that appear in the view query and any queries referenced indirectly from other views).
- The VSO must have SELECT privilege on all tables referenced in the view.

Required Ownership for FedSQL Views

Objects such as tables and views do not have owners in SAS Federation Server, so ownership is granted on the schema containing the view. For example,

```
ALTER SCHEMA LD_ORA1_SERVICE.TKTSTST1 OWNER TO USER1
```

A definer's rights view must be associated with a schema owner. If an invoker calls a definer view for which the schema has no owner, an error similar to the following is returned:

```
ERROR: Definer's security context for view "%.*s" cannot be established because
the schema container "%.*s"."%.*s" has no configured owner.
```

For additional information about schema ownership, see “Catalogs and Schemas” on page 88.

Creating FedSQL Views

Overview

Views are created using the CREATE VIEW statement or command. You can create views from a single data source or multiple data sources. To create a view the user must have the **CREATE VIEW** privilege on the view schema, or inherited from a parent object. The CREATE VIEW privilege is not necessary for users to create views on schemas where the user is the owner of the schema.

Here is the syntax to create a FedSQL view for a single data source using invoker's rights. For a definer rights view, replace INVOKER with DEFINER:

```
CREATE VIEW MYVIEW SECURITY INVOKER AS SELECT * FROM
CAT1.S1.MYTABLE T1
```

Create a FedSQL View from Multiple Data Sources

You can create a FedSQL view across multiple data sources. Suppose that data resides in two separate data sources, one in Oracle and the other in DB2. Using CREATE VIEW, you can tie the two data sources together to create a single federated view of the data.

To create a FedSQL view across two data sources:

1. Create a data service for each of the data sources that you want to access. Make note of the catalog names associated with each of the data sources. In this example, CAT1 and CAT2 are the catalog names. Also make note of the schema within the catalog where your data resides.

   ```
   Data source one: T1 CAT1.S1.MYTABLE
   Data source two: T2 CAT2.S2.MYOTHERTABLE
   ```

2. Invoke the SQL Console window and connect to one or more data services on SAS Federation Server. The data service can be one of the two created above or any other data service associated with the SAS Federation Server that you are using. Normally, you would use a DSN to connect to the data service.
3. Create a statement using Submit:

   ```
   CREATE VIEW MYVIEW SECURITY
   INVOKER AS SELECT * FROM CAT1.S1.MYTABLE
   T1, CAT2.S2.MYOTHERTABLE T2 WHERE T1.X=T2.X
   ```

   This statement creates an invoker's rights view. For definer's rights view, replace INVOKER with DEFINER.
4. Grant **SELECT** privileges to the users or groups that will access the view.

All users with permissions can now read from the view.

Dynamic Connections

Overview

A dynamic connection is a connection made during the execution of a FedSQL view. Dynamic connections are created within the initial set of connections when a user connects to SAS Federation Server but does not include a connection to data referenced within a view. The dynamic connection feature allows an administrator to create views that reference data from any data service defined in SAS Federation Server, but does not require the user's DSN to reference all the data sources in the view.

For example, an administrator creates a view V1, which references data in catalog C1 and C2, where C1 and C2 were defined through separate data services. Without dynamic connections, the administrator would need to create a federated DSN that included a connection to:

- The data service where V1 was stored.
- The data service containing catalog C1.
- The data service containing catalog C2.

Assume the view definition in V1 was changed to reference additional data in catalogs C3 and C4, each coming from a different data service. Without dynamic connections, the administrator would need to modify the user DSN(s) to include references to the data services containing catalogs C3 and C4.

Dynamic connections ease this administration burden because the administrator can simply create a user DSN to include a connection to the data service(s) containing the views. Any data required by the view is accessed through connections made dynamically during view execution. If the view definitions change, the user DSNs do not need to change. This feature is also very useful with data caching, as the data cache can be moved from one data service to another without requiring modifications to user DSNs.

Dynamic connections are transient, so they do not modify the capabilities of the user's original connection properties. Dynamic connections only occur within the context of FedSQL views (invoker's or definer's rights). They do not occur any other place in the system.

Object Privileges and Required Logins

For dynamic connections to function properly, the invoker or definer, depending on the view type, must have CONNECT privileges on the data service that is referenced by the

dynamic connection. Privileges on the DSN are insufficient for dynamic connections to succeed because the underlying connection is made to the data service, and not through the DSN.

Example: USER1 connects across 2 different DSNs: BASEDSN in data service BASE and ORADSN in data service ORA1. USER1 then creates an invoker's rights view that references tables in both DSNs, and stores the view in BASEDSN. USER2 then connects to BASEDSN and issues a 'select * from VIEW'. The view will succeed only if USER2 has CONNECT privileges on the ORA1 data service.

Also, note that the invoker or definer of the view must have SELECT privileges on any data referenced in the view. Also, the invoker or definer must have any required logins to the data sources that require dynamic connections. For example, if an invoker's rights view requires a dynamic connection to reference data from catalog ORACAT in an Oracle data service, the invoker must have a login to the Oracle database in order to make the connection. The login can either be a personal or shared login.

Secured Connections

Connections made dynamically during view execution are secured. If a connection is made to an Oracle service known by the ORACAT catalog, and a FedSQL view is read (**`select * from ORACAT.schema.view`**), and the view query references another catalog (**`select * from DB2CAT.schema.table`** ...), then the dynamic connection to the DB2 service known by DB2CAT is secured. This is true even if the ORACAT connection is unsecured.

For example, if you connect via DSN=ORACLE_DSN, where the DSN is unsecured but configured to use the FEDSQL driver, the DSN might expand to a connection string like this:

```
DRIVER=FEDSQL;conopts=(driver=ORACLE;catalog=ORACAT;...)
```

The data that lives under ORACAT is accessed without additional SAS Federation Server authorization enforcement applied.

If an additional SELECT statement is made:

```
SELECT* from ORACAT.Schema.View.
```

No SELECT privilege check is made against the columns of **`ORACAT.Schema.View`**.

Now assume that the view content is just a simple indirection that expands to this:

```
SELECT* from DB2CAT.Schema.Table ...
```

If DB2CAT's data service is configured to use the ODBC driver, the server will attempt to dynamically connect to DB2CAT using a secured connection that is equivalent to:

```
DRIVER=FEDSQL;conopts=(driver=ODBC;catalog=DB2CAT;...;
odbc_dsn=DB2DSN)
```

Data Caching

Overview

Data caching can be used to manage the performance of frequently accessed data sources to minimize impact on the databases and operational servers. You can free up resources for the high-availability data sources by caching data that is used often and fairly

constant. Overall, data caching can have a positive effect on user satisfaction and system performance.

When query optimization alone is not sufficient, caching provides an alternative with greater flexibility than traditional replication and consolidation techniques. Any FedSQLdefiner's rights view can be used to create a cache and caches can be refreshed periodically to remain synchronized with their parent views. Queries can be processed against caches just as if you were accessing the original data source.

A definer's rights view can be cached in any catalog defined in SAS Federation Server, with the exception of any catalog defined in an MDS service. At the time the cache is refreshed, the view is executed and the results are stored in a table which is named and maintained by SAS Federation Server. When a user selects from the cached view, results are returned from the data stored in the table, and not from a dynamic execution of the view. This improves performance by eliminating the need to derive a new FedSQL execution plan, fetching data from possibly slow or unavailable data sources, and performing SQL operations such as joins or function evaluations. Instead, the view execution merely fetches the data stored in the table created during the cache refresh.

Data caching can have a positive impact on server performance. You can use data caching to pre-calculate results. Using the following example, a SELECT from view V1 would fetch data and calculate the results, outputting a single number. If you cache the result anyone selecting from the view will receive the result immediately. For example,

```
CREATE VIEW V1 AS SELECT AVG(B1) FROM A,B,C,D,E
where E .C1 < (SELECT AVG(C1) FROM E) AND B.C1 = A.C1 AND
C.C1 < B.C1 AND D.C1 * E.C1 > SUM(B.C1)
```

You can also cache tables or result sets so that they remain consistent during multiple queries. For example, an ORDERS table is updated continuously during the day as customers purchase products. Caching the data guarantees consistent results and reduces the load on the servers, freeing resources to process incoming orders.

Views and Caching

SAS Federation Server through the use of FedSQL, allows users to cache data from a definer's rights view, creating a materialized view of the data, also known as the cache table. A cache table is a snapshot of the target view from a specific point in time.

Only definer's rights views can be used to cache data. To recap, when a definer's rights view is executed, it uses the credentials of the view’s schema owner rather than those of the current user to access catalogs that are referenced in the view. A definer's rights view returns the same result from the underlying database, no matter who is requesting the data, thereby allowing a single copy of the view result set to be cached and consumed by all users. SAS Federation Server authorization, including table, column and row-level security, can be used to provide a granular and user-specific access to the view.

If a definer’s rights view is altered to an invoker’s rights view, the view is no longer cached.

Before you begin, ensure that the following prerequisites and configuration tasks have been addressed for both definer’s rights views and cached views.

Requirements for Cached Views

Note: This section uses the following acronyms:

- **VS** (View Schema) is the schema where the cached view resides.
- **CS** (Cache Schema) is the schema where the cache tables reside.

- **VSO** (View Schema Owner) is the owner of the schema where the cached view resides.
- **CSO** (Cache Schema Owner) is the owner of the schema where the cache tables reside.

Here are the requirements for definer's rights views that are cached:

- A user that is allowed to create or drop cached views within a schema should be granted CREATE CACHE privilege on the VS or an object in its inheritance hierarchy. This privilege also allows the user to refresh the cache.
- A user that is allowed to refresh cached views within the schema should be granted ALTER CACHE privilege on the VS, unless they already have CREATE CACHE privilege. ALTER CACHE also allows the user to enable or disable a cache.
- A user that is creating a cache also needs the CONNECT privilege on the ADMIN DSN.
- Only definer's rights views can be cached.
- Administrators implicitly have all privileges, including CREATE CACHE and ALTER CACHE.
- A schema owner must be assigned to the CS object. This is the cache table schema owner, also referred to as the CSO. The CSO effectively owns all cache tables in its schema, and this identity is used for executing the view and saving the cached data.
- The CSO must have CONNECT privilege on the data service the cache tables are in.
- The CSO must have a database login to the database of the cached tables if the cached location requires credentials. This can be a personal login or a shared login. This is needed for cache creation and refresh.
- The CSO must have a database login to the database of the cached view. This can be a personal login or a shared login. The server impersonates the CSO user during cache creation and refresh, and the CSO must be able to select from the original view.

  Note: During data cache connections, the CSO connects to the databases that contain the CS and the VS using a Credential Search Order of "PERSONAL, SHARED". See "Configuring DSNs" on page 89 for more information.
- The VSO must be granted CREATE TABLESPACE privilege on the CS for the cache tables. This privilege allows the server to cache results sets in the CS for views owned by the VSO. The server assumes the identity of the CSO user to create and drop cached tables in the CS, to insert and delete rows in the cache table, and to select data from the cache table during client access.

Working with Cached Views

Overview

You can configure cached views using one of these methods:

- Issue administration DDL statements such as CREATE CACHE, REFRESH CACHE, ALTER CACHE, and DROP CACHE.
- Use the Data Cache module in SAS Federation Server Manager.

Administration DDL statements are described in Appendix 1. Procedures for caching data in SAS Federation Server Manager are described in the application's online help.

The following scenarios describe how data operations are performed using various DDL statements.

Creating a Cache

Use the CREATE CACHE DDL statement on page 178 cache a definer's rights view or change an existing cache definition. The CREATE CACHE privilege is required on the view that is being cached.

A cache table is created when the CREATE CACHE statement is executed. A cache table is also created when the ALTER CACHE statement is executed with the REFRESH option. A cache table is a snapshot of the target view from a specific point in time.

The CREATE CACHE statement adds entries to the OBJECTS and CONFIG_OBJECTS information views. Several options are available with the CREATE CACHE statement.

Alter a Cache

To alter an existing cache, use the ALTER CACHE DDL statement on page 180 with available options REFRESH, ENABLE and DISABLE.

To refresh an existing cached view, use ALTER CACHE with the REFRESH option. The REFRESH option creates a new cache table which is a snapshot of the target view at the time when the refresh is done. The required privileges are CREATE CACHE or ALTER CACHE on the view that is being cached or refreshed.

The following example refreshes an existing cached view using the most recent definition:

```
ALTER CACHE  [view_catalog_na me.[view_schema_name.]]view_name REFRESH
```

You can also disable and enable a cached view using ALTER CACHE. For additional details, see “Disabling and Enabling Cached Views” on page 102 .

Purge Cache

Purge Cache forces the removal of outdated cache tables that are no longer in use. Only system users or those with ADMINISTER privilege can execute this DDL statement.

There are two commands that you can use to purge cache tables:

- Use the explicit PURGE CACHE DDL statement on page 181 to activate the cache table cleanup process for all cache tables created through SAS Federation Server. When PURGE CACHE is issued, messages are returned indicating the cache tables that were successfully removed and what problems were encountered. This command has no options.

  ```
  PURGE CACHE
  ```

- Schedule cache table cleanup for certain intervals using the ALTER SERVER statement. The syntax to set the time-out is:

  ```
  ALTER SERVER {OPTIONS(xset PURGE_CACHE XX)}
  ```

  where ***xx*** is the time-out value in minutes.

  - A negative value indicates that the cleanup thread will wake only when the PURGE CACHE command is issued. It never wakes up automatically.
  - A value of 0 indicates that the cleanup thread will wake whenever a CREATE CACHE, ALTER CACHE REFRESH, DROP CACHE, or PURGE CACHE

statement is issued, or when the view is dropped. Note that this might not clean up all old caches since some might still be in use.

- A positive value indicates how often (in minutes) the cleanup thread will wake-up to remove orphaned cache tables.

Note: Cleanup is not run on deferred caches, when CREATE CACHE includes an option value of [DEFERRED].

Drop Cache

Use the DROP CACHE DDL statement on page 181 or issue a DROP VIEW command to drop a cache. Invoking DROP VIEW also drops all of the view's associated cache tables. DROP CACHE requires CREATE CACHE privilege on the view.

```
DROP CACHE [view_catalog_name.[view_schema_name.]]view_name [FORCE]
DROP VIEW [view_catalog_name.[view_schema_name.]]view_name [FORCE]
```

Disabling and Enabling Cached Views

Overview

In the event that a cached view needs to be taken offline for any reason, it can be temporarily disabled. Disabling a cached view does not drop or delete a cached view. Instead, the cached view is temporarily suspended while the users are rerouted to the original definer's rights view that the cached view was built on. At the time the cached view is enabled, users are transparently directed back to the actual cached view.

Disabling a Cached View

Use the ALTER CACHE DDL statement on page 180 with the DISABLE option to disable a cached view. ALTER CACHE requires the CREATE CACHE or ALTER CACHE privilege on the view.

The following example disables an existing cached view and redirects users to the original definer's rights view while the cached view is offline:

```
ALTER CACHE [view_catalog_name.[view_schema_name.]]view_name  DISABLE
```

When the cached view is disabled, the original definer's rights view is used. While the current cached view remains disabled but continues to be reported with a status of suspended, the disabled cache view displays the following behavior:

- An ALTER SERVER REFRESH refreshes the cached view but does not enable the cached view. It remains disabled with a status of suspended.
- A CREATE CACHE behaves normally and the cached view remains disabled in a suspended status.
- An ALTER CACHE ENABLE re-enables the cached view and drops the suspended status.
- An ALTER CACHE DISABLE on a cached view that is disabled in a suspended status, returns a success message.
- An ALTER CACHE ENABLE on a cached view that is not disabled also returns a success message.

Enabling a Cached View

Use the ALTER CACHE DDL statement on page 180 with the ENABLE option to reinstate a disabled cached view. Once a cached view is enabled, users are redirected

from the original definer's rights view to the actual cached data. ALTER CACHE requires the CREATE CACHE or ALTER CACHE privilege on the view.

The following example enables a cached view that is currently disabled and brings the cached data online:

```
ALTER CACHE [view_catalog_name.[view_schema_name.]]view_name ENABLE
```

Memory Data Store

Overview

Memory Data Store, or MDS, is a transactional in-memory data store that can be used with SAS Federation Server. The MDS driver supports most of the FedSQL functionality. MDS runs strictly in memory with no backup data store. Therefore, changes are lost when the database is dropped or the server is restarted.

The database is created in memory when the first user connects to the database. The database remains in memory until one of the following conditions is met:

- The server is shut down.
- The data service or the catalog associated with the data service is dropped.

Note: You cannot drop a MDS data service or catalog if users are connected to the data service.

You can rename the database and change the memory value while users are connected, but you cannot drop the database while users are connected. To drop or rename a schema, the table within the schema cannot not be in use. Users can be connected to the database, but they cannot have a table open in the schema.

The MDS Data Service

You can configure an MDS data service and table using one of these methods:

- Use the CREATE DATA SERVICE DDL statement on page 163.
- Use the **New Data Service** function in SAS Federation Server Manager.

You can create multiple data services if needed.

Catalog and Schema Support

Overview

Each MDS data service catalog contains a pre-defined, read-only schema, named SYSTEMINFO. The SYSTEMINFO schema contains an auto-generated MEMORY table that is described in the following section.

Limiting the Database Memory Size

To limit the memory size for a database, use the connection string option MAXDBMEM= that specifies the maximum size of memory to be used to store all rows of data in the database. This includes committed rows and pending row versions (INSERT, UPDATE, and DELETE operations that have not yet been committed or rolled back). If an INSERT, UPDATE, or DELETE operation exceeds this limit, an out

of memory error is returned. For additional information, see Connection Options in the "MDS Driver Reference" on page 122.

The MEMORY Table

The MEMORY table contains information about memory usage and is always available. The first row contains statistics about the database. Additional rows provide information about each of the tables in the database.

The MEMORY table includes the following columns:

Table 6.1 *Columns in SYSTEMINFO.MEMORY Table*

Column Name	Description
"DB_NAME"	The name of the current database. This will be the same as the catalog name.
"SCHEMA_NAME"	The name of the schema for the table (NULL for the database info row).
"TABLE_NAME"	The name of the table (NULL for the database info row).
"ROW_COUNT"	The number of rows in the table (NULL for the database info row).
"ROW_SIZE"	The size of a single row in the table (NULL for the database info row).
"MEM_SIZE"	The current memory used by the table and database for data.
"MEM_PEAK"	The peak memory used by the table and database since creation.
"MEM_LIMIT"	The maximum memory this database can use (NULL for table info rows). This value corresponds to the MAXDBMEM= option specified when the database was created.

To view the MEMORY table using SAS Federation Server Manager:

1. In the navigation tree, open the MDS data service.
2. Select the catalog associated with the MDS data service.
3. Select the SYSTEMINFO schema to open it. The MEMORY table appears under the SYSTEMINFO schema.

Understanding Data Federation and Best Practices

Overview

Successful data federation projects require careful preparation and attention in two areas:

- The data model
- Data security

Proper understanding of your organization's data access needs is key to a successful data federation deployment. The data model will control the type of underlying work required at run time to satisfy requests for data. A poorly constructed data model can result in inefficient performance and perhaps incorrect results if the data is not well understood. Data security design should consider how your users will gain access to the back-end data. Will users be accessing the data under their individual authorizations? Will you establish one or more data owners who will act as data access proxies for the end users? There are several options to consider in the design phase so that SAS Federation Server can be properly configured.

Data Model

A good starting place is to identify your data sources, understand the relationships between different sets of data, and derive a data model that meets the needs of your business. This is the same type of background work that typically goes into a data warehousing project, where the end result is often a set of tables or views that are loaded in a data mart. But unlike data warehousing, with SAS Federation Server, the data does not need to be copied from the source into a data mart. Instead, the data can be fetched from their source locations and operated on in real time during the data requests. These operations need to be as efficient as possible, and this will require a well-thought data model.

Data caching should be considered when designing the data model. It can provide a vital role in optimizing query performance by pre-executing and storing intermediate results. This can be particularly useful for back-end data sources that have low availability, slow access speeds, expensive access fees, or contain data where real time values are not required. A data model can consist of a set of user-visible FedSQL views that are dependent on other restricted FedSQL views which can be dependent on back-end data (for example, tables and database views). FedSQL views are very similar to relational database views, except that the data can come from a heterogeneous set of data sources.

Any combination of FedSQL views can be cached, though the views must be definer's rights views. The cached views can be refreshed on a periodic basis through the scheduler, or refreshed at any time through direct administration SQL commands. If certain data sets will be used frequently or involve complicated SQL operations to return the results, you will want to consider data caching for those results. Cached results can be joined with uncached data to provide quick responses to user queries.

Another consideration is to use the Memory Data Store (MDS) to store frequently used or temporary data. MDS allows tables to be stored in the memory of the server process. This allows for extremely fast data access performance. These tables must be managed manually and are automatically deleted when the server is shut down. FedSQL can join an MDS table with tables from any other data source, whether they are cached or uncached, including other MDS tables.

Data Security

Overview

In a federated system, data can be acquired from a large variety of data sources, and your users might not always have direct access to those systems. Even if they do, administering a large number of database credentials and setting up database privileges such that all users can access all data with their personal credentials can be burdensome. The Federation Server security capabilities provide some alternatives which can help ease security administration.

Invoker's Rights Views

If your users already have direct access to the back-end systems and you want them to access the data under their individual or shared logins, then you can configure SAS Federation Server to access the data under the rights of the invoking user. This is done by creating invoker's rights FedSQL views. When these views execute, any connections to back-end data will require that the invoking user have proper credentials to access the data. In addition, you will want to ensure that SAS Federation Server security settings allow invokers to access the required data. For example, if a FedSQL view is defined to select data from an Oracle™ data source and a DB2™ data source, the administrator will need to ensure that proper privileges are assigned to each view invoker on the Oracle™ and DB2™ data, in addition to the FedSQL view itself. The privileges can be granted to a group, or set of groups that the invokers are members of, instead of to each individual invoker.

Because the user must be able to directly access all of his data, you will need to secure each object for each user. And invoker's rights views need to be sensitive to these security settings on underlying objects, particularly column level security. Consider a view that selects columns C1 and C2 from table T, for example,

```
SELECT C1, C2 FROM T
```

If table T has been secured through the Federation Server such that User1 does not have SELECT privilege on the table, then when User1 selects from the view, he will get an error as it attempts to access table T. You might consider denying SELECT privilege on the view to User1 in this case. Furthermore, if User2 has SELECT privilege on table T1 and column C1 but not on column C2, then User2 will get an error when selecting from the view. Putting column level security on the view to disallow SELECT privilege on column C2 to User2 will not help, since the underlying view definition specifically requires access to column C2 in table T. When creating views that require data from other objects that have column-level security, you might consider using the '*' to select all columns, for example:

```
SELECT * FROM T
```

When the Federation Server is configured with **SelectStarExpansion=VISIBLE**, the **'*'** will expand to all columns for which the invoking user has SELECT privilege. This enables you to create a single view usable by all invokers, yet each invoker will see the columns for which he has SELECT privilege only.

Definer's Rights Views

If your users do not have direct access to the back-end systems, or you want to read objects under the authorizations of a single user, then you can configure the Federation Server to access the data under the rights of the defining user. This is done by creating definer's rights FedSQL views. When these views are executed by any user, any connections to back-end data will be done under the definer of the view (the definer is actually the schema owner containing the view).

In this security model, you will generally deny access to the back-end data to all users except the view definer. Any column-, table- or row-level security will be set on the view itself. For example, if your view is defined as:

```
SELECT C1, C2 FROM T
```

You will want to ensure that the view definer has full access to columns C1 and C2 in table T. All other users do not require access to back-end data. And in many cases, if your users are interfacing only with exposed top-level objects of the data model, then you can deny privileges to those users on back-end objects. Individual Federation Server security settings can then be consolidated on just the exposed objects of the data model (usually FedSQL views). Users then access only the views; no other credentials are

required. This greatly simplifies the security settings in the Federation Server and reduces administration on all the back-end data sources as well.

Mixed Models

Note that you can use a combination of approaches with both invoker's and definer's rights views intermingled. FedSQL views can be nested, and view types are honored. For example, you might create view V1 that is owned by Owner1 and data is accessed through the credentials of Owner1. However, if view V1 selects from view V2 which is another definer's rights view but owned by Owner2, then all data within view V2 is accessed under the authorizations of Owner2. If view V2 selects from view V3 which is an invoker's rights view, then any data within view V3 continues to be accessed by Owner2, who is the invoker of the view.

Dropping Objects

SAS Federation Server persists metadata for security settings in its set of system tables. The server attempts to synchronize this metadata with the actual objects that it represents. For example, if the administrator has granted SELECT privilege on table T1 to User1, and subsequently table T1 is dropped, the server will likewise drop its security metadata onto T1. If T1 is subsequently re-created, it will have no security set on the object itself. All security definitions for T1 will come from its inheritance objects, including the schema, catalog, data service and server.

If you prefer that the security metadata is retained on the server, there are a couple suggested possibilities. First, it might be that a job is dropping the table only to re-create it in a subsequent step. This is often the case when refreshing the contents of a table. Rather than dropping and re-creating the table (which often has the additional side-effect of losing any indexes on the table), consider issuing a DELETE FROM TABLE command to delete all the rows from the table. Here, the underlying table still remains intact, as do the security metadata stored in the system tables.

A second approach is to use definer's rights views to assist with individual object security. Often times, data in back-end data sources is being manipulated (created and dropped) by a few "power" users only. These users will require CREATE, DROP TABLE, and VIEW privileges. The CREATE privilege set applies only to container objects (schema, catalog, data service, and server) and not individual tables or views. The DROP privilege can be applied at the object level, but if designed correctly, can be used from the container object.

More often, business users are the ones requiring table and column level privileges to access the data. If you choose the security model using definer's rights views, then you can place your individual privileges on the views, which do not go away when a back-end table is dropped and re-created. Back-end data can be secured by denying privileges on the container object to business users, but granting SELECT to view owners. An approach similar to this can be used to eliminate the need for table-, column- and row-level security on back-end data source objects.

Chapter 7

Driver Reference for SAS Federation Server

Database Functionality and Driver Performance

Because SAS Federation Server supports several data sources, a broad range of database functionality that is unique to each data source is provided. For example, a particular data source provides transaction support while another data source might not provide transaction support but supports indexes and integrity constraints.

You must understand database functionality and how its implementation affects processing, performance, and integrity of your data in order to determine which data sources are most appropriate for different types of applications. Because database functionality is unique to each data source, you cannot make assumptions about the data source to be accessed. For example, an application cannot request a locking level just because that locking level is more efficient. An application must respond to the attributes of a SAS Federation Server driver.

Database functionality is applied through the SAS Federation Server driver when the application submits requests, which can be in the form of FedSQL statements or the SQL statements that are the implementation of the data service. For additional information about database functionality including data types, see the*SAS FedSQL Reference Guide*.

DB2 Reference

Understanding the SAS Federation Server Driver for DB2

The SAS Federation Server Driver for DB2 enables SAS Federation Server to read and update legacy DB2 tables. In addition, the driver creates DB2 tables that can be accessed by both SAS Federation Server and the DB2 database management system (DBMS).

The SAS Federation Server Driver for DB2 supports most of the FedSQL functionality. The driver also supports an application's ability to submit native DB2 SQL statements.

The SAS Federation Server Driver for DB2 is a remote driver, which means that it connects to a server process in order to access data. The process might be running on the same machine as SAS Federation Server, or it might be running on another machine in the network.

Connection Options for DB2

Overview

To access data that is hosted on SAS Federation Server, a client must submit a connection string, which defines how to connect to the data. The data service connection arguments for a DB2 data set include connection options and advanced options.

Connection Options

Connection options are used to establish a connection to a data source. Specify one or more connection options when defining a data service using the CREATE DATA SERVICE DDL statement on page 163 .

The following connection options are supported for DB2 data sources.

Option	Description
CATALOG	**CATALOG=*catalog-identifer*;** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid (for example, **catalog=DB2**). *Note:* You must specify a catalog. For the DB2 database, this is a logical catalog name to use as an SQL catalog identifier.
DATABASE|DB	**DATABASE=*database-specification*;** Specifies the name of the DB2 database, for example, **database=sample, DB=sample**. *Note:* You must specify a database name.
DRIVER	**DRIVER=DB2;** Identifies the DB2 data source to which you want to connect. *Note:* You must specify the driver.

Advanced Options

The following advanced options are supported for DB2 data sources.

Option	Description
CLIENT ENCODING	**CLIENT_ENCODING=*encoding-value*** Used to specify the encoding of the DB2CODEPAGE to the DB2 driver. When using this option, you must also set the DB2CODEPAGE environment variable on the client. When the encoding of the DB2 client layer (stored in DBCODEPAGE) is different from the encoding value of the DB2 operating system value, which is generally the SAS® session encoding value, the DB2 client layer attempts to convert incoming data to the DB2 encoding value that is stored in DB2CODEPAGE. You must first determine the correct value for DB2CODEPAGE and then set the CLIENT_ENCODING= option to match the corresponding encoding value in DB2CODEPAGE in order to prevent the client layer from converting the data incorrectly. For example, suppose you are storing Japanese characters in a DB2 database and the client machine where the DB2 driver is executing is a Windows® machine running CP1252 encoding. When the application tries to extract the data into <ph conref="Resources/mcvars.dita#mcvars/variableId5" />, the DB2 client layer attempts to convert these Japanese characters into Latin1 representation, which does not contain Japanese characters. As a result, a garbage character appears to indicate a failure in transcoding. To resolve this situation, you must first set the DB2CODEPAGE environment variable value to 1208 (the IBM® code page value that matches UTF-8 encoding) to specify that the DB2 client layer send the data to the application in UTF-8 instead of converting it into Latin1. In addition, you must specify the corresponding encoding value of DB2CODEPAGE because the SAS Federation Server Driver for DB2 cannot derive this information from a DB2 session. For this particular Windows case, set the CLIENT_ENCODING= option to the UTF-8 to match the DB2CODEPAGE value (1208) in order to specify the DB2CODEPAGE value to the DB2 driver. However, changing the value of DB2CODEPAGE affects all applications that run on that machine. You should reset the value to the usual DB2CODEPAGE value, which was derived when the database was created. *Note:* Setting the DB2CODEPAGE value or the CLIENT_ENCODING= value incorrectly can cause unpredictable results. You should set these values only when a situation such as the example above occurs. *Note:* You can specify any valid encoding value for CLIENT_ENCODING=option.
CT_PRESERVE	**CT_PRESERVE=STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.

Option	Description
DEFAULT_ATTR	**DEFAULT_ATTR=(*attr=value;...*)** DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **attr=value** corresponds to any of the options below: • **CURSORS=*n*-** Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **DEFAULT_ATTR=(CURSORS=2)** • **USE_EVP=*n*—** Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **DEFAULT_ATTR=(USE_EVP=0)** • **XCODE_WARN=*n*** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **DEFAULT_ATTR=(XCODE_WARN=1)**
DRIVER TRACE	**DRIVER_TRACE='API | SQL | ALL';** Requests tracing information, which logs transaction records to an external file that can be used for debugging purposes. The SAS Federation Server driver writes a record of each command that is sent to the database to the trace log based on the specified tracing level, which determines the type of tracing information. The tracing levels are: • **ALL** Activates all trace levels. • **API** Specifies that API method calls be sent to the trace log. This option is most useful if you are having a problem and need to send a trace log to Technical Support for troubleshooting. • **DRIVER** Specifies that driver-specific information be sent to the trace log. • **SQL** Specifies that SQL statements that are sent to the database management system (DBMS) be sent to the trace log. Tracing information is DBMS specific, but most SAS Federation Server drivers log SQL statements such as SELECT and COMMIT. **Default:** Tracing is not activated. *Note:* If you activate tracing, you must also specify the location of the trace log with DRIVER_TRACEFILE=. Note that DRIVER_TRACEFILE= is resolved against the TRACEFILEPATH set in ALTER SERVER. TRACEFILEPATH is relative to the server's Content Root location. (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=. **Interaction:** You can specify one trace level, or you can concatenate more than one by including the | (OR) symbol. For example: **driver_trace='api|sql'** generates tracing information for API calls and SQL statements.

Option	Description
DRIVER TRACE FILE	**DRIVER_TRACEFILE=*'filename'***; Used to specify the name of the text file for the trace log. Include the filename and extension in single or double quotation marks. For example: **driver_tracefile='\mytrace.log'** **Default:** The default TRACEFILE location applies to a relative filename, and it will be placed relative to TRACEFILEPATH. **Requirement:** DRIVER_TRACEFILE is required when activating tracing using DRIVER_TRACE. **Interaction:** (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=.
DRIVER TRACE OPTIONS	**DRIVER_TRACEOPTIONS=APPEND** | **THREADSTAMP** | **TIMESTAMP;** Specifies options in order to control formatting and other properties for the trace log: • **APPEND** Adds trace information to the end of an existing trace log. The contents of the file are not overwritten. • **THREADSTAMP** Prepends each line of the trace log with a thread identification. • **TIMESTAMP** Prepends each line of the trace log with a time stamp. **Default:** The trace log is overwritten with no thread identification or time stamp.
PASSWORD	**PWD=*password*** Specifies the password for DB2..
USER ID	**UID=*user-id*;** Specifies the DB2 login user ID.

FedSQL Driver Reference

Overview

The FedSQL language driver supports the FedSQL dialect, as documented in the *Federation Server FedSQL Language Reference Guide*. When loaded, the FedSQL driver parses SQL requests, and then sends the parsed query to the appropriate SAS Federation Server driver to determine whether the functionality can be handled by the data service. The FedSQL driver itself includes an SQL processor which supports the FedSQL dialect. The main emphasis of the FedSQL driver is to support federation of data sources. For example, if an SQL submission is requesting data from DB2 to be joined with data from Oracle, the SQL processor will request the data from the data sources and then perform the join itself in SAS Federation Server. The FedSQL driver supports the FedSQL dialect over any data source. So even if the SQL request is from a single data source, if that data source does not support the particular SQL functionality, the FedSQL processor will guarantee its implementation.

The FedSQL driver is also required for SAS Federation Server SQL Authorization Enforcement. If the DSN is configured to enable Federation Server SQL Authorization Enforcement, then the FedSQL driver is automatically loaded and used. The FedSQL

dialect can also be requested when creating a DSN by choosing the FedSQL dialect for the DSN.

The FedSQL driver supports various connection options, as outlined below. You must use the DEFAULT_ATTR option in the CREATE DSN statement to specify these options. For example,

```
CREATE DSN MYDSN UNDER "Oracle Service" CONNECT
'DEFAULT_ATTR=(SQL_MAX_COL_SIZE=500);DRIVER=ORACLE'
```

Connection Options

- DEFAULT_CATALOG=*catalog-name* - Used to specify the name of the catalog to set as the current catalog upon connecting. This option is useful for SQL Server connections and federated connections.
- DEFAULT_ATTR=(*attr=value*;...) - Used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **`attr=value`** corresponds to any of the following options:

SQL_CURSORS=*n*

FedSQL connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2.

- A value of 0 causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one.
- A value of 1 causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used.
- A value of 2 (default) causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available, otherwise the cursor will be forward only.

```
DEFAULT_ATTR=(SQL_CURSORS=2)
```

SQL_AC_BEHAVIOR=*n*

FedSQL connection handle option. Specifies whether FedSQL should use transactions when processing complex operations. For example, **`"CREATE TABLE xxx AS SELECT yyy FROM zzz"`** or a multi-row delete statement that requires multiple operations to delete the underlying rows. Possible values are 0 (default), 1 and 2.

- A value of 0 (default) means that no transactions are attempted under-the-covers and operations such as emulated UDPATE, DELETE or INSERT.
- A value of 1 means that FedSQL tries to use transaction to better support the correct behavior when AUTOCOMMIT is ON (where individual operations like UPDATE, DELETE and INSERT should be atomic).
- A value of 2 means that transactions are required. This option will fail if the underlying drivers do not support transactions.

```
DEFAULT_ATTR=(SQL_AC_BEHAVIOR=0)
```

SQL_MAX_COL_SIZE=*n*

FedSQL statement handle option. Allows a user to specify the size of the varchar or varbinary that is used for the potentially truncated long data when direct bind is not possible.

- The default value is 32767.
- The limit for this size is 1 MG. If the value exceeds 1 MG, FedSQL resets the value and returns an **`Option value changed`** warning.

```
DEFAULT_ATTR=(SQL_MAX_COL_SIZE=1048576)
```

SQL_PUSHDOWN=*n*

FedSQL statement handle option. This option tells FedSQL if and when it should try to push down SQL to the underlying driver. The values are 8, 2 or 0 (default).

- A value of 8: (PLAN_FORCE_PUSHDOWN_SQL) - Complete statement pushdown is required. If that is not possible, then the INSERT, UPDATE, DELETE, or CREATE TABLE AS statement should fail.
- A value of 2: (PLAN_DISABLE_PUSHDOWN_SQL) - Specifies that the INSERT, UPDATE, DELETE, or CREATE TABLE AS statement not be pushed down to the underlying driver.
- A value of 0 (default): Specifies that the FedSQL processor decides whether the INSERT, UPDATE, DELETE, or CREATE TABLE AS statement should be pushed down to the underlying driver.

```
DEFAULT_ATTR=(SQL_PUSHDOWN=0)
```

SQL_STMT_MEM_LIMIT=*n*

FedSQL statement handle option. Used to control the amount of memory available to FedSQL to answer SQL requests.

- (*n*)umber is treated as an integer and is specified in bytes.
- The following example allows 200 MB of memory:

```
DEFAULT_ATTR=(SQL_STMT_MEM_LIMIT=209715200)
```

SQL_TXN_EXCEPTIONS=*n*

FedSQL connection handle option. Supports dynamic connections regardless of the specified transaction isolation. Possible values are 0 or 2 (default).

- Specify a value of 0 to disable support for dynamic connections.
- Specify a value of 2 to enable support for dynamic connections.

```
DEFAULT_ATTR=(SQL_TXN_EXCEPTIONS=2)
```

SQL_USE_EVP=*n*

FedSQL statement handle option. This option optimizes the driver for large result sets. The possible values are 0 or 1 (default) and are used as follows:

- Specify 0 to turn optimization OFF.
- Specify 1 to enable optimization (ON).

```
DEFAULT_ATTR=(SQL_USE_EVP=0)
```

SQL_VDC_DISABLE=*n*

FedSQL statement handle option. This option is used to allow or disallow use of cached data for a statement. The possible values are 0 (default) or 1 and are used as follows::

- Specify a value of 0 to enable cached data.
- Specify a value of 1 to disable cached data.

```
DEFAULT_ATTR=(SQL_VDC_DISABLE=1)
```

SQL_XCODE_WARN=*n*

FedSQL statement handle option. Used to warn if there is an error while transcoding data during row input or output operations. Possible values are 0 (default), 1 or 2 and are used as follows::

- Specify 0 to return an error if data cannot be transcoded.
- Specify 1 to return a warning if data cannot be transcoded.
- Specify 2 to ignore transcoding errors.

```
DEFAULT_ATTR=(SQL_XCODE_WARN=1)
```

Greenplum Reference

Understanding the Driver for Greenplum

The SAS® Federation Server Driver for Greenplum enables SAS Federation Server to read and update legacy Greenplum tables. In addition, the driver creates Greenplum tables that can be accessed by both SAS Federation Server and Greenplum.

The SAS® Federation Server Driver for Greenplum supports most of the FedSQL functionality. The driver also supports the application's ability to submit native Greenplum SQL statements.

The SAS® Federation Server Driver for Greenplum is a remote driver, which means that it connects to a server process in order to access data. The process might be running on the same machine as SAS Federation Server, or it might be running on another machine in the network.

Connection Options for Greenplum

Connection Options

Connection options are used to establish a connection to a data source. Specify one or more connection options when defining a data service using the CREATE DATA SERVICE DDL statement on page 163 .

The following connection options are supported for the Greenplum database.

Option	Description
CATALOG	**CATALOG=*catalog-identifier*;** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid (for example, **catalog=gps_test**). *Note:* You must specify a catalog. For the Greenplum database, this is a logical catalog name to use as an SQL catalog identifier.
DATABASE	**DATABASE=*database name*;** Identifies the database to which you want to connect, which resides on the server previously specified through the SERVER option.
DRIVER	**DRIVER=GREENPLUM**; Identifies the data service to which you want to connect, which is a Greenplum database. *Note:* You must specify the driver.
DSN	**DSN=*data_source_identifer*;** Identifies the data source name to which you want to connect.
SERVER	**SERVER=*server_name*;** Identifies the name of the server where the Greenplum database resides.

Advanced Options

The following advanced options are supported for the Greenplum database.

Option	Description
ALLOW UNQUOTED NAMES	**ALLOW_UNQUOTED_NAMES=NO** | **YES**; Specifies whether to enclose table and column names with quotation marks. Tables and columns are quoted when this option is set at NO (default). If set to YES, the driver will not automatically add quotation marks to table and column names if they are not specified. This allows Greenplum tables and columns to be created in the default lower case.
CLIENT ENCODING	**CLIENT_ENCODING=cei**; Specifies an encoding different from the default to use on the client side.

Option	Description
CT_PRESERVE	**`CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE`** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.
DEFAULT_ATTR	DEFAULT_ATTR=(*attr*=*value*;...) DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **`attr=value`** corresponds to any of the options below: • **`CURSORS=n`**- Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **`DEFAULT_ATTR=(CURSORS=2)`** • **`USE_EVP=n`**— Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **`DEFAULT_ATTR=(USE_EVP=0)`** • **`XCODE_WARN=n`** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **`DEFAULT_ATTR=(XCODE_WARN=1)`**

Option	Description
DRIVER TRACE	**DRIVER_TRACE='API | SQL | ALL';** Requests tracing information, which logs transaction records to an external file that can be used for debugging purposes. The SAS Federation Server driver writes a record of each command that is sent to the database to the trace log based on the specified tracing level, which determines the type of tracing information. The tracing levels are: • **ALL** Activates all trace levels. • **API** Specifies that API method calls be sent to the trace log. This option is most useful if you are having a problem and need to send a trace log to Technical Support for troubleshooting. • **DRIVER** Specifies that driver-specific information be sent to the trace log. • **SQL** Specifies that SQL statements that are sent to the database management system (DBMS) be sent to the trace log. Tracing information is DBMS specific, but most SAS Federation Server drivers log SQL statements such as SELECT and COMMIT. **Default:** Tracing is not activated. *Note:* If you activate tracing, you must also specify the location of the trace log with DRIVER_TRACEFILE=. Note that DRIVER_TRACEFILE= is resolved against the TRACEFILEPATH set in ALTER SERVER. TRACEFILEPATH is relative to the server's Content Root location. (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=. **Interaction:** You can specify one trace level, or you can concatenate more than one by including the | (OR) symbol. For example: **driver_trace='api|sql'** generates tracing information for API calls and SQL statements.
DRIVER TRACEFILE	**DRIVER_TRACEFILE='*filename*';** Used to specify the name of the text file for the trace log. Include the filename and extension in single or double quotation marks. For example: **driver_tracefile='\mytrace.log'** **Default:** The default TRACEFILE location applies to a relative filename, and it will be placed relative to TRACEFILEPATH. **Requirement:** DRIVER_TRACEFILE is required when activating tracing using DRIVER_TRACE. **Interaction:** (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=.
DRIVER TRACE OPTIONS	**DRIVER_TRACEOPTIONS=APPEND | THREADSTAMP | TIMESTAMP;** Specifies options in order to control formatting and other properties for the trace log: • **APPEND** Adds trace information to the end of an existing trace log. The contents of the file are not overwritten. • **THREADSTAMP** Prepends each line of the trace log with a thread identification. • **TIMESTAMP** Prepends each line of the trace log with a time stamp. **Default:** The trace log is overwritten with no thread identification or time stamp.
MAX_BINARY_LEN	**MAX_BINARY_LEN=*value*;** Specifies a value to limit the length of long binary fields (LONG VARBINARY). As opposed to other databases, Greenplum does not have a size limit for long binary fields.

Option	Description
MAX_CHAR_LEN	**MAX_CHAR_LEN=*value*;** Specifies a value to limit the length of character fields (CHAR and VARCHAR). As opposed to other databases, Greenplum does not have a size limit for character fields.
MAX_TEXT_LEN	**MAX_TEXT_LEN=*value*;** Specifies a value to limit the length of long character fields (LONG VARCHAR). As opposed to other databases, Greenplum does not have a size limit for long character fields.
NUM BYTES PER CHAR	**NUMBYTESPERCHAR=*value*;** Specifies the default number of bytes per character.
PASSWORD	**PASSWORD=*password*;** Specifies a password for the ID passed through the USER= option. The alias is PWD=. *Note:* You must specify the PASSWORD= option.
SCHEMA	**SCHEMA=*value*;** Specifies the default schema for the connection. If not specified, the schema (or list of schemas) will be determined based on the value of the schema search path defined on the database server.
STRIP BLANKS	**STRIP_BLANKS=*value*;** Specifies whether to strip blanks from character fields.
USER	**USER=*user-id*;** Specifies a Greenplum user ID. If the ID contains blanks or national characters, enclose it in quotation marks. The alias is UID=. *Note:* You must specify the USER= option.

Greenplum Wire Protocol Driver Usage Notes

SAS Federation Server provides a number of wire protocol ODBC drivers that communicate directly with a database server, without having to communicate through a client library. When you configure the ODBC drivers on Windows or UNIX, you have the opportunity to set certain options. SAS products run best when these options are selected. Some, but not all, are selected by default.

Windows	The options are located on the **Advanced** or **Performance** tabs in the ODBC Administrator.
UNIX	The options are available when configuring data sources using the **dfdbconf** tool. Values can also be set by editing the odbc.ini file in which their data sources are defined.

Note: For DSNs where a wire protocol driver is specified and the catalog option is selected, only the schemas that have tables or views will be listed (not all schemas

that exist). If you require different behavior, create a DSN without the catalog option.

When configuring an ODBC DSN using the Greenplum 64-bit Wire Protocol driver, select the following options on the **Advanced** tab:

- **Enable SQLDescribeParam**
- **Fetch Ref Cursor**
- **Application Using Threads**

MDS Driver Reference

About the MDS Driver

The MDS driver provides access to an in-memory database for SAS Federation Server.

Data Types

MDS supports the following data types. All types support NULL.

Data Type	Description
NUMERIC	precision scale
DECIMAL	precision scale
DATE	'yyyy/mm/dd'
TIME	'hh:mm:ss'
TIMESTAMP	'yyyy/mm/dd hh:mm:ss[.ssss]'
BINARY(len)	fixed length binary data
VARBINARY(maxlen)	variable length binary data
CHAR(len) [CHARACTER SET xxx]	fixed length character data
VARCHAR (maxlen) [CHARACTER SET xxx]	variable length character data
NCHAR(maxlen)	fixed length Unicode data
NVARCHAR(maxlen)	variable length Unicode data
BIGINT	64-bit signed integer
UBIGINT	64-bit unsigned integer

Data Type	Description
INTEGER	32-bit signed integer
UINTEGER	32-bit unsigned integer
DOUBLE	8-byte IEEE floating-point values (missing values are not supported)

Connection Options

MDS supports the following connection string options.

Option	Description
CATALOG	**CATALOG=*catalog name***; Specifies the catalog name.
NUMERICS	**NUMERICS= Y | N** Allows numeric data types or treats them as double precision. The default is Y (Yes).
CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.
IDCASE	**IDCASE=SENSITIVE | INSENSITIVE** Specifies if schema, table, column, and alias identifiers are case-sensitive or insensitive. The default is case sensitive.

Option	Description
DEFAULT_ATTR	**DEFAULT_ATTR=(*attr=value;...*)** DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **attr=value** corresponds to any of the options below: • **CURSORS=*n*** - Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **DEFAULT_ATTR=(CURSORS=2)** • **USE_EVP=*n*** — Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **DEFAULT_ATTR=(USE_EVP=0)** • **XCODE_WARN=*n*** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **DEFAULT_ATTR=(XCODE_WARN=1)**
DEFSCHEMA	**DEFSCHEMA=*schema name*** Specifies the default schema for identifiers with no schema qualifier. The default is the first SCHEMA= in the connection string.
REFTYPE	**REFTYPE=VARCHAR | NVARCHAR | VARBINARY** Indicates that duplicate column data should be stored once and referenced by result sets rather than having separate instances in each row. This reduces memory usage with large numbers of duplicate data but might slow down performance. • **VARCHAR** Create a REFCHAR instead of a VARCHAR when specified. The default is create VARCHAR. • **NVARCHAR** Create an NREFCHAR instead of an NVARCHAR when specified. The default is create NVARCHAR. • **VARBINARY** Create a REFBINARY instead of a VARBINARY when specified. The default is create VARBINARY. *Note:* A REFCHAR(32) uses less space than a VARCHAR(32) if there are many duplicate values in the table or if the data is less than 32 characters. However, a REFCHAR(1) generally uses more memory than a VARCHAR(1) because an extra pointer has to be stored instead of a single character.
MAXDBMEM	**MAXDBMEM=number of bytes** Specifies the maximum amount of memory the database can use to store all row data for all tables. The default is no limit.

ODBC Reference

About ODBC

This section provides functionality details and guidelines for the open database connectivity (ODBC) databases that are supported by the SAS Federation Server Driver for ODBC.

ODBC standards provide a common interface to a variety of databases, including dBASE, Microsoft® Access®, Oracle®, Paradox, and Microsoft SQL Server databases. Specifically, ODBC standards define APIs that enable an application to access a database if both the application and the database conform to the specification. ODBC also provides a mechanism to enable dynamic selection of a database that an application is accessing, so that users have the flexibility of selecting databases other than those that are specified by the application developer.

Understanding the Driver for ODBC

The SAS Federation Server Driver for ODBC enables SAS Federation Server to read and update legacy ODBC database tables. In addition, the driver creates tables that can be accessed by both SAS Federation Server and an ODBC database.

The SAS Federation Server Driver for ODBC supports most of the FedSQL functionality. The driver also supports an application's ability to submit native database-specific SQL statements.

The SAS Federation Server Driver for ODBC is a remote driver, which means that it connects to a server process in order to access data. The process might be running on the same machine as SAS Federation Server, or it might be running on another machine in the network.

Connection Options for ODBC

Overview

To access data that is hosted on SAS Federation Server, a client must submit a connection string, which defines how to connect to the data. The data service connection arguments for an ODBC-compliant database include connection options and advanced options.

Connection Options

Connection options are used to establish a connection to a data source. Specify one or more connection options when defining a data service using the CREATE DATA SERVICE DDL statement on page 163 .

The following connection options are supported for an ODBC-compliant database:

Option	Description
CATALOG	**CATALOG=*catalog-identifier*;** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. For databases that do not support native catalogs, any identifier is valid (for example, catalog=myodbc). For the SQL Server, which is a multi-catalog database, CATALOG= is not required. The connection defaults to CATALOG=* unless you specify a logical name for the catalog and map it to the native catalog name in the SQL Server. For example, to map the logical catalog mycat to the native catalog named newusers, specify the following code: **catalog=(mycat=newusers);** *Note:* You are required to specify a catalog name for all databases except the SQL Server, which can omit the CATALOG= option, or else specify a catalog name map. Catalog name maps can be used with FedSQL only. They cannot be used to access native DBMS SQL.
CONOPTS	**CONOPTS=(*ODBC–compliant database connection string*);** Specifies, within parentheses, an ODBC-compliant database connection string, including any valid SAS Federation Server Driver for ODBC connection options that are not provided by SAS Federation Server connection options. For connections that do not use a DSN, include the ODBC-specific DRIVER= keyword, that is, if you do not include a DSN= connection option to specify an ODBC DSN. For example, specify DRIVER={SQL Server} followed by appropriate connection options that are valid in SQL Server connections. If you include a DSN= or FILEDSN= specification with the CONOPTS= option, do not use the ODBC_DSN= connection option. However, you can specify the ODBC database-specific connection options by using CONOPTS= and then you can specify an ODBC DSN that contains other connection information by using the ODBC_DSN= connection option.
DRIVER	**DRIVER=ODBC**; Calls the SAS Federation Server Driver for ODBC. This specifies that the data service to which you want to connect must be an ODBC-compliant database. *Note:* You must specify the driver.
ODBC_DSN	**ODBC_DSN=*odbc dsn name*** Specifies a valid ODBC-compliant database DSN that contains connection information for connecting to the ODBC-compliant database. You can use the CONOPTS= option in addition to ODBC_DSN= option to specify database-specific connection options not provided by SAS Federation Server. Do not specify the ODBC DSN in both CONOPTS= and ODBC_DSN=.

Advanced Options

conref="Resources/Snippets/advanced_option.dita#advanced_option/snippet

The following connection options are supported for an ODBC-compliant database:

Option	Description
CT_PRESERVE	**`CT_PRESERVE = STRICT	SAFE	FORCE	FORCE_COL_SIZE`** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.
ENABLE MULTIPLE ACTIVE RESULT SETS (MARS)	**`ENABLE_MARS= 0	1`** Enables or disables the use of multiple active result sets (MARS) on SQL Server. FedSQL cannot permit transactions on top of SQL Server because SQL Server only allows one cursor per transaction. Set this option to 1 (true) which gives FedSQL the ability to allow transactions under a given SQL Server connection.

Option	Description
DEFAULT_ATTR	**DEFAULT_ATTR=(*attr=value;...*)** DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **attr=value** corresponds to any of the options below: • **CURSORS=*n*** - Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **DEFAULT_ATTR=(CURSORS=2)** • **USE_EVP=*n***— Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **DEFAULT_ATTR=(USE_EVP=0)** • **XCODE_WARN=*n*** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **DEFAULT_ATTR=(XCODE_WARN=1)**

Option	Description
DEFAULT CURSOR TYPE	**DEFAULT_CURSOR_TYPE=FORWARD_ONLY | KEYSET_DRIVEN | DYNAMIC | STATIC;** Specifies a valid default cursor type for new statements. The valid options are: • **FORWARD_ONLY**Specifies a non-scrollable cursor that moves only forward through the result set. Forward-only cursors are dynamic in that all changes are detected as the current row is processed. If an application does not require scrolling, the forward-only cursor retrieves data quickly, with the least amount of overhead processing. • **KEYSET_DRIVEN**Specifies a scrollable cursor that detects changes that are made to the values of rows in the result set but that does not always detect changes to deletion of rows and changes to the order of rows in the result set. A keyset-driven cursor is based on row keys, which are used to determine the order and set of rows that are included in the result set. As the cursor scrolls the result set, it uses the keys to retrieve the most recent values in the table. It is sometimes helpful to have a cursor that can detect changes in the rows of a result set. A keyset-driven cursor uses a row identifier rather than caching the entire row into memory. It therefore uses much less disk space than other row caching mechanisms. Deleted rows can be detected when a SELECT statement that references the bookmark, row ID, or key column values fails to return a row. • **DYNAMIC**Specifies a scrollable cursor that detects changes that are made to the rows in the result set. All INSERT, UPDATE, and DELETE statements that are made by all users are visible through the cursor. The dynamic cursor is good for an application that must detect all concurrent updates that are made by other users. • **STATIC**Specifies a scrollable cursor that displays the result set as it existed when the cursor was first opened. The static cursor provides forward and backward scrolling. If the application does not need to detect changes but requires scrolling, the static cursor is a good choice. **Default:** There is no default value. *Note:* The application can still override this value, but if the application does not explicitly set a cursor type, this value will be in effect
DRIVER TRACE	**DRIVER_TRACE='API | SQL | ALL';** Requests tracing information, which logs transaction records to an external file that can be used for debugging purposes. The SAS Federation Server driver writes a record of each command that is sent to the database to the trace log based on the specified tracing level, which determines the type of tracing information. The tracing levels are: • **ALL** Activates all trace levels. • **API** Specifies that API method calls be sent to the trace log. This option is most useful if you are having a problem and need to send a trace log to Technical Support for troubleshooting. • **DRIVER** Specifies that driver-specific information be sent to the trace log. • **SQL** Specifies that SQL statements that are sent to the database management system (DBMS) be sent to the trace log. Tracing information is DBMS specific, but most SAS Federation Server drivers log SQL statements such as SELECT and COMMIT. **Default:** Tracing is not activated. *Note:* If you activate tracing, you must also specify the location of the trace log with DRIVER_TRACEFILE=. Note that DRIVER_TRACEFILE= is resolved against the TRACEFILEPATH set in ALTER SERVER. TRACEFILEPATH is relative to the server's Content Root location. (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=. **Interaction:** You can specify one trace level, or you can concatenate more than one by including the | (OR) symbol. For example: **driver_trace='api|sql'** generates tracing information for API calls and SQL statements.

Option	Description
DRIVER TRACE FILE	**DRIVER_TRACEFILE='*filename*';** Used to specify the name of the text file for the trace log. Include the filename and extension in single or double quotation marks. For example: **driver_tracefile='\mytrace.log'** **Default:** The default TRACEFILE location applies to a relative filename, and it will be placed relative to TRACEFILEPATH. **Requirement:** DRIVER_TRACEFILE is required when activating tracing using DRIVER_TRACE. **Interaction:** (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=.
DRIVER TRACE OPTIONS	**DRIVER_TRACEOPTIONS=APPEND | THREADSTAMP | TIMESTAMP;** Specifies options in order to control formatting and other properties for the trace log: • **APPEND** Adds trace information to the end of an existing trace log. The contents of the file are not overwritten. • **THREADSTAMP** Prepends each line of the trace log with a thread identification. • **TIMESTAMP** Prepends each line of the trace log with a time stamp. **Default:** The trace log is overwritten with no thread identification or time stamp.
PASSWORD	**PASSWORD=*password*;** Specifies the password that corresponds to the user ID in the database. *Note:* The alias is **PWD=**.
USER	**USER=*user-ID*;** Specifies the user ID for logging on to the ODBC-compliant database, such as Microsoft SQL Server, with a user ID that differs from the default ID. *Note:* The alias is **UID=**.

The following examples are of valid connection strings.

This connection string specifies an ODBC DSN:

```
driver=odbc; uid=scott; pw=roger; odbc_dsn=myOracleDSN;
    catalog=odbc_oracle;
```

This connection string specifies catalog name maps to access multiple catalogs on Microsoft SQL Server:

```
driver=odbc; uid=jfox; pw=mypw; odbc_dsn=mySQLdsn;
    catalog=(cat1=mycat; cat2=testcat; cat3=users;
```

Wire Protocol Driver Usage Notes

SAS Federation Server provides a number of wire protocol ODBC drivers that communicate directly with a database server, without having to communicate through a client library. When you configure the ODBC drivers on Windows or UNIX, you have the opportunity to set certain options. SAS products run best when these options are selected. Some, but not all, are selected by default.

Windows	The options are located on the **Advanced** or **Performance** tabs in the ODBC Administrator.
UNIX	The options are available when configuring data sources using the **dfdbconf** tool. Values can also be set by editing the odbc.ini file in which their data sources are defined.

Note: For DSNs where a wire protocol driver is specified and the catalog option is selected, only the schemas that have tables or views will be listed (not all schemas that exist). If you require different behavior, create a DSN without the catalog option.

When configuring an ODBC DSN using the 32-bit MySQL Wire Protocol driver, select the following options on the **Advanced** tab:

- **Application Using Threads**
- **Enable SQLDescribeParam**

Configure the following **Advanced** options for the SQL Server Legacy Wire Protocol driver:

- **Enable Quoted Identifiers**
- **Fetch TWFS as Time**
- **Fetch TSWTZ as Timestamp**

Note:

1. Significant performance improvements have been realized when using the SQL Server Legacy Wire Protocol Driver, as compared to the SQL Server Wire Protocol Driver.
2. The SQL Server Legacy Wire Protocol Driver does not support transactions when used with FedSQL enabled because the driver only allows a single statement per connection while FedSQL requires multiple statements per connection when using transactions.

Oracle Reference

Understanding the SAS Federation Server Driver for Oracle

The SAS Federation Server Driver for Oracle enables SAS Federation Server to read and update legacy Oracle tables. In addition, the driver creates Oracle tables that can be accessed by both SAS Federation Server and Oracle.

The SAS Federation Server Driver for Oracle supports most of the FedSQL functionality. The driver also supports the application's ability to submit native Oracle SQL statements.

The SAS Federation Server Driver for Oracle is a remote driver, which means that it connects to a server process in order to access data. The process might be running on the same machine as SAS Federation Server, or it might be running on another machine in the network.

Connection Options for Oracle

Overview

To access data that is hosted on SAS Federation Server, a client must submit a connection string, which defines how to connect to the data. The data service connection arguments for Oracle Server include connection options and advanced options.

Connection Options

Connection options are used to establish a connection to a data source. Specify one or more connection options when defining a data service using the CREATE DATA SERVICE DDL statement on page 163 .

The following connection options are supported for Oracle Server:

Option	Description
CATALOG	**CATALOG=*catalog identifier*;** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid (for example, catalog=oracle_test). *Note:* You must specify a catalog. For the Oracle database, this is a logical catalog name to use as an SQL catalog identifier.
DRIVER	**DRIVER=ORACLE**; Identifies the data service to which you want to connect, which is an Oracle database. *Note:* You must specify the driver.
PATH	**PATH=*database-specification*;** Specifies the Oracle connect identifier. A connect identifier can be a net service name, a database service name, or a net service alias.
USERID (UID)	**UID=*user-id*;** Specifies an optional Oracle user ID. If the user ID contains blanks or national characters, enclose it in quotation marks. If you omit an Oracle user ID and password, the default Oracle user ID OPS$sysid is used, if it is enabled. UID= must be used with PWD=.
PASSWORD (PWD)	**PWD=*password*;** Specifies an optional Oracle database password that is associated with the Oracle user ID. PWD= is always used with UID= and the associated password is case-sensitive. If you omit PWD=, the password for the default Oracle user ID OPS$sysid is used, if it is active.

Advanced Options

The following advanced options are supported for Oracle Server:

Option	Description
CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.
DEFAULT_ATTR	**DEFAULT_ATTR=(*attr=value*;...)** DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **attr=value** corresponds to any of the options below: • **CURSORS=*n*-** Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **DEFAULT_ATTR=(CURSORS=2)** • **USE_EVP=*n***— Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **DEFAULT_ATTR=(USE_EVP=0)** • **XCODE_WARN=*n*** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **DEFAULT_ATTR=(XCODE_WARN=1)**

Option	Description
DRIVER TRACE;	**DRIVER_TRACE='API | SQL | ALL'** Requests tracing information, which logs transaction records to an external file that can be used for debugging purposes. The SAS Federation Server driver writes a record of each command that is sent to the database to the trace log based on the specified tracing level, which determines the type of tracing information. The tracing levels are: • **ALL** Activates all trace levels. • **API** Specifies that API method calls be sent to the trace log. This option is most useful if you are having a problem and need to send a trace log to Technical Support for troubleshooting. • **DRIVER** Specifies that driver-specific information be sent to the trace log. • **SQL** Specifies that SQL statements that are sent to the database management system (DBMS) be sent to the trace log. Tracing information is DBMS specific, but most SAS Federation Server drivers log SQL statements such as SELECT and COMMIT. **Default:** Tracing is not activated. *Note:* If you activate tracing, you must also specify the location of the trace log with DRIVER_TRACEFILE=. Note that DRIVER_TRACEFILE= is resolved against the TRACEFILEPATH set in ALTER SERVER. TRACEFILEPATH is relative to the server's Content Root location. (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=. **Interaction:** You can specify one trace level, or you can concatenate more than one by including the | (OR) symbol. For example: **driver_trace='api|sql'** generates tracing information for API calls and SQL statements.
DRIVER TRACEFILE	**DRIVER_TRACEFILE='*filename*';** Used to specify the name of the text file for the trace log. Include the filename and extension in single or double quotation marks. For example: **driver_tracefile='\mytrace.log'** **Default:** The default TRACEFILE location applies to a relative filename, and it will be placed relative to TRACEFILEPATH. **Requirement:** DRIVER_TRACEFILE is required when activating tracing using DRIVER_TRACE. **Interaction:** (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=.
DRIVER TRACE OPTIONS	**DRIVER_TRACEOPTIONS=APPEND | THREADSTAMP | TIMESTAMP;** Specifies options in order to control formatting and other properties for the trace log: • **APPEND** Adds trace information to the end of an existing trace log. The contents of the file are not overwritten. • **THREADSTAMP** Prepends each line of the trace log with a thread identification. • **TIMESTAMP** Prepends each line of the trace log with a time stamp. **Default:** The trace log is overwritten with no thread identification or time stamp.
ORA_ENCODING	**ORA_ENCODING=UNICODE;** Specifies that the Oracle data be returned in Unicode to SAS Federation Server. UNICODE is the default setting and is independent of the NLS_LANG environment variable setting.

Option	Description
ORNUMERIC	**ORANUMERIC=NO | YES** Specifies how numbers read from or inserted into the Oracle NUMBER column will be treated. This option defaults to YES so a NUMBER column with precision or scale is described as TKTS_NUMERIC. This option can be specified as both a connection option and a table option. When specified as both connection and table option, the table option value overrides the connection option. • **NO** Indicates that the numbers will be treated as TKTS_DOUBLE values. They might not have precision beyond 14 digits. • **YES** Indicates that non-integer values with explicit precision will be treated as TKTS_NUMERIC values. This is the default setting.
USE CACHED CATALOG	**USE_CACHED_CATALOG=YES | NO;** Specifies whether to use the cached catalog rather than compiling a new catalog on every run. Setting this option to YES can improve the performance of the TKTSForeignKeys API. The default setting is YES. *Note:* Before you can use this option, you must complete the following steps: 1. Create a materialized view. See the example code in "Creating a Materialized View (USE_CACHED_CATALOG)" on page 136. 2. Use the ALTER DSN statement to add the USE_CACHED_CATALOG connection option. For more information about the ALTER DSN statement, see ALTER DSN Statement.

Creating a Materialized View (USE_CACHED_CATALOG)

The following example shows you how to create a materialized view. Use this script if USE_CACHED_CATALOG is set to YES above.

```
/*----------------------SAS_CACHED_CATALOG.SQL-------------------------------*/
/* This script is used to create the materialized and the synonym needed to
   get the ForeignKey metadata.  Work with your DBA to set this up.
   Materialized views can be complex and so thorough understanding will help us
   use them effectively. Especially deciding how to do the refreshes.
   Here we provide the simplest possible steps to create the required materialized
   view and the command to refresh it manually. The materialized view below can
   be created in any schema with any name. Feel free to add whatever REFRESH
   options suits your purpose. Note that you might need additional steps based
   on the REFRESH option setting. Here we provide the simplest possible way to
   do this.  The PUBLIC synonym pointing to this Materialized view must be
   named "SAS_CACHED_FK_CATALOG_PSYN". This synonym must be visible to
   PUBLIC (or the set of users who will be needing Foreignkey metadata) so that
   it is accessible from any   schema.
*/

Create materialized view SAS_CACHED_FK_CATALOG_MATVIEW REFRESH ON DEMAND as SELECT
PKAC.OWNER as PKTABLE_SCHEM,
PKAC.TABLE_NAME as PKTABLE_NAME,
PKACC.COLUMN_NAME as PKCOLUMN_NAME,
FKAC.OWNER as FKTABLE_SCHEM,
FKAC.TABLE_NAME as FKTABLE_NAME,
FKACC.COLUMN_NAME as FKCOLUMN_NAME,
FKACC.POSITION as KEY_SEQ,
FKAC.CONSTRAINT_NAME as FK_NAME,
PKAC.CONSTRAINT_NAME as PK_NAME
from
sys.all_constraints PKAC, sys.all_constraints FKAC,
sys.all_cons_columns PKACC, sys.all_cons_columns FKACC

where

FKAC.r_constraint_name=PKAC.constraint_name and
FKAC.constraint_name=FKACC.constraint_name and
PKAC.constraint_name=PKACC.constraint_name and PKAC.constraint_type='P' and
FKAC.constraint_type='R' and FKAC.owner=FKACC.owner and PKAC.owner=PKACC.owner
and PKAC.table_name=PKACC.table_name and FKAC.table_name=FKACC.table_name and
FKACC.position = PKACC.position ;

/* The synonym name *must* be SAS_CACHED_FK_CATALOG_PUBLIC_SYNONYM */
create public synonym  SAS_CACHED_FK_CATALOG_PSYN for SAS_CACHED_FK_CATALOG_MATVIEW;
grant all on  SAS_CACHED_FK_CATALOG_PSYN to PUBLIC;

/*---------Manual REFRESH of the Materialized View-----------------------------*/
/* Note there are several ways to do this, consult with your DBA.
   Here are a couple of ways:
*/
execute DBMS_MVIEW.REFRESH('SAS_CACHED_FK_CATALOG_MATVIEW');
execute DBMS_SNAPSHOT.REFRESH('SAS_CACHED_FK_CATALOG_MATVIEW', '?');
```

Oracle Wire Protocol Driver Usage Notes

SAS Federation Server provides a number of wire protocol ODBC drivers that communicate directly with a database server, without having to communicate through a client library. When you configure the ODBC drivers on Windows or UNIX, you have the opportunity to set certain options. SAS products run best when these options are selected. Some, but not all, are selected by default.

Windows	The options are located on the **Advanced** or **Performance** tabs in the ODBC Administrator.
UNIX	The options are available when configuring data sources using the **`dfdbconf`** tool. Values can also be set by editing the odbc.ini file in which their data sources are defined.

Note: When you use a wire protocol driver to create an ODBC connection, the following special considerations apply:

1. For DSNs where a wire protocol driver is specified and the catalog option is selected, only the schemas that have tables or views will be listed (not all schemas that exist). If you require different behavior, create a DSN without the catalog option.
2. Verify that the Enable Bulk Load option is turned on in the ODBC DSN for those databases that support this option. The Bulk Load option is not on by default in the newer wire protocol drivers. If the Bulk Load option is not on, then insert performance suffers.

When configuring an ODBC DSN using the Oracle Wire Protocol driver (32–bit and 64-bit), set the following options.

On the **Advanced** tab in the ODBC Setup window, select the following options:

- **Application Using Threads**
- **Enable SQLDescribeParam**
- **Describe at Prepare**
- **Enable N-CHAR Support**

On the **Performance** tab in the ODBC Setup window, select the **Enable Scrollable Cursors** option.

SAP Reference

Understanding the SAS Federation Server Driver for SAP

The SAS Federation Server Driver for SAP enables SAS Federation Server to read tables from SAP systems. The SAS Federation Server Driver for SAP has read-only capabilities.

The SAS Federation Server Driver for SAP supports most of the FedSQL functionality. The driver does not support the application's ability to submit native SQL statements.

The SAS Federation Server Driver for SAP is a remote driver, which means that it connects to a server process in order to access data. The process might be running on the same machine as SAS Federation Server, or it might be running on another machine in the network.

Data Service Connection Arguments for SAP

The following table describes the data service connection arguments for SAP.

Connection Argument
ABAPFM = abap_function_name Specifies the name of the Advanced Business Application Programming (ABAP) function module that the driver uses internally. • **Syntax:** APBAPFM | ABAPFUNCTION | ABAPFUNC = *abap_function_name* • **Default:** /SAS/Z_SAS_DIALOG
ABAP_NAMESPACE = namespace Specifies the namespace for ABAP functions and programs that are used by the driver. If the ABAP programs are installed in the customer namespace rather than in the default namespace, this parameter identifies where the ABAP programs are installed. • **Syntax:** ABAP_NAMESPACE | ABAPNAMESPACE | ABAP_NAME_SPACE | ABAPNS | ABAP_NS = *namespace* • **Default:** /SAS/
ABAPPROG = abap_program Specifies the name of the ABAP language that the driver uses internally. This value is set by the ABAP function module. • **Syntax:** ABAPPROG | ABAPREPORT | ABAPPROGRAM = *abap_program* • **Default:** /SAS/Z_SAS_READ
ASHOST = application_server_host Specifies the host name of the server or IP address of a specific application server. • **Syntax:** ASHOST | HST | RFCHOST | R3HOST = *application_server_host* • **Default:** None.
BATCH = 0 | 1 | Y | N Specifies whether the SAS Federation Server Driver for SAP should use SAP batch jobs for the data extracts. • **Y** The SAS Federation Server Driver for SAP uses batch jobs to extract R/3 data. • **N** The SAS Federation Server Driver for SAP uses dialog processes to extract R/3 data. • **Syntax:** BATCH | BATCH_MODE | BATCHMODE = 0 | 1 | Y | N • **Default:** N
BUFFER_SIZE = buffersize Sets the minimum buffer size for data transfers in batch and dialog modes. The number of bytes should be greater than 10,000 and no more than eight digits. • **Syntax:** BUFFERSIZE | BUFFSIZE | BLOCK_SIZE | BLOCKSIZE | BUFFER_SIZE = *buffersize* • **Default:** 100,000 bytes

Connection Argument

CLIENT = client

Specifies the SAP logon parameter client. Examples for a client are 000 or 800.

- **Syntax:** CLIENT | CLI | RFCCLIENT | RFCCLI = *client*
- **Default:** SAP system default

Note: When you access the SAP system via the driver, specify valid logon information including client, user name, password, and language. The user ID and password might also be provided through single sign-on (SSO). The driver performs a logon check at OPEN time.

DEFAULT_ATTR=(*attr=value*)

Used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where attr=value corresponds to any of the options below:

- CURSORS=*n*— FedSQL connection handle option. This option controls the driver's use of client side result set cursors. the possible values are 0, 1 and 2.

  0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one.

  1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used.

  2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only

- USE_EVP=*n*- FedSQL statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default.
- XCODE_WARN=*n*- FedSQL statement handle options. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore).

DESTGROUP = destination_group

Specifies the name of the destination group for batch access to the SAP system. The destination groups are defined in the /SAS/DESTS table in SAP.

- **Syntax:** DESTGROUP = *destination_group*
- **Default:** SAS1

DESTINATION = destination

Specifies the destination in the saprfc.ini file, if working with a saprfc.ini file. If the RFC server is an R/3 system, this destination must also be defined in the SIDEINFO file for the SAP gateway.

- **Syntax:** DESTINATION | DEST| DST | DSTN = *destination*
- **Default:** None.

GROUP = application_server_group

Specifies the name of the group of application servers, if load balancing is being used.

- **Syntax:** GROUP = *application_server_group*
- **Default:** None.

Connection Argument

GWHOST = gateway_host_name

Specifies the host name of the SAP gateway, if the server is R/2 or external.

- **Syntax:** GWHOST |GATEWAY_HOST = *gateway_host_name*
- **Default:** None.

GWSERV = gateway_service

Specifies the service of the SAP gateway, if the server is an R/2 server or external.

- **Syntax:** GWSERV| GATEWAY_SERVICE = *gateway_service*
- **Default:** None.

IEEE_REVERSE = Y | N

Specifies whether floating point numbers are byte reversed.

- **Y:** Specifies that floating-point numbers are byte reversed.
- **N:** Specifies that floating-point numbers are not byte reversed.
- **Syntax:** IEEE_REVERSE = Y | N
- **Default:** For an R/3 application server on Windows NT, the value is Y. On other platforms, the value is N.

INENCODING = code_page

Specifies the code page. Indicates the code page of the SAP server. The encoding is determined by the value returned by the SAP server. In some rare cases, it might be necessary to override this value by setting the inencoding = connection parameter.

- **Syntax:** INENCODING = *code_page*
- **Default:** None.

LANGUAGE = language

Specifies the SAP logon parameter language. The value for language is either the 2-byte ISO-language key or the 1-byte SAP-language. Examples for the language are EN, DE or E, D.

- **Syntax:** LANGUAGE | LANG | LNG |RFCLANG |RFCLNG = *language*
- **Default:** SAP system default

Note: When you access the SAP system via the driver, specify valid logon information including client, user name, password and language. The user ID and password might also be provided through SSO. The driver performs a logon check at OPEN time.

MAX_TABLE_JOINS = number

Specifies the number of tables that can be used in a left outer join or an inner join in ABAP Open SQL.

- **Syntax:** MAX_TABLE_JOINS | MAX_TABLES_JOIN | MAX_TABLES_JOINS | MAX_TABLE_JOIN = *number*
- **Default:** 25

MSHOST = message_server_host

Specifies the host name of the Message Server, if load balancing is being used.

- **Syntax:** MSHOST= *message_server_host*
- **Default:** None.

Connection Argument

PWD = password

Specifies the SAP logon parameter password.

- **Syntax:** PASSWORD | PASSWD | PWD | PW | PASS = *password*
- **Default:** None.

Note: When you access the SAP system via the driver, specify valid logon information including client, user name, password and language. The user ID and password might also be provided through SSO. The driver performs a logon check at OPEN time.

R3NAME = system_name

Specifies the name of the R/3 system, if load balancing is being used.

- **Syntax:** R3NAME = *system_name*
- **Default:** None.

RFC_STRING = additional_rfc_options

Specifies additional logon or connection parameters for the RfcOpenEx() call. The SAS Federation Server Driver for SAP uses the RfcOpenEx() call to log on to the SAP system. Using this option, parameters which are not SAS Federation Server Driver for SAP connection attributes can be passed to the RfcOpenEx call. For example: **`RFC_STRING = "ABAP_DEBUG=1"`**

This option can be used to support future extensions of the RfcOpenEx call.

- **Syntax:** RFC_STRING | RFCSTRING | RFC_OPTIONS_EXT | RFCOPENEX | ADDITIONAL_RFC_OPTIONS = *additional_rfc_options*
- **Default:** None.

SAPLOGON_ID = saplogon_id

Specifies the string defined for SAPLOGON on a Windows 32-bit system.

- **Syntax:** SAPLOGON_ID = *saplogon_id*
- **Default:** None.

SYSNR = system_number

Specifies the SAP system number, if load balancing is not being used. The number is the two-byte code that identifies the system on the host, for example, 00 and 01.

- **Syntax:** SYSNR |SYS |SYSTEM |SYSNO = *system_number*
- **Default:** None.

TRACE = 0 | 1 | Y | N

Specifies if the SAS Federation Server Driver for SAP should trace requests. If the trace option is switched on, the driver writes log information into a file. The RFC library logs messages in the dev_rfc file.

- **0 or N:** RFC trace is switched off.
- **1 or Y:** RFC trace is switched on.
- **Syntax:** TRACE = 0 | 1 | Y | N
- **Default:** 0

Note: The RFC trace directory is set in the RFC_TRACE_DIR environment variable.

Connection Argument
UID = user Specifies the SAP logon parameter user. • **Syntax:** USER | USR | RFCUSER | USERNAME | USERID = *user* • **Default:** None. *Note:* When you access the SAP system via the driver, specify valid logon information including client, user name, password, and language. The user ID and password might also be provided through SSO. The driver performs a logon check at OPEN time.

Installing and Configuring the SAS Federation Server Driver for SAP

Overview

Installing the Driver for SAP involves several steps that you must complete in the appropriate sequence. Review the system requirements and authorization profiles, set up the SAS Federation Server Driver for SAP, and then install the SAP® components on the SAP system and SAS Federation Server.

After installing the SAS Federation Server Driver for SAP, you must complete additional steps to configure it to be used by SAS Federation Server.

SAS Federation Server Driver for SAP Requirements

The following list identifies the system requirements for a SAS Federation Server Driver for SAP that are used by SAS Federation Server. After the driver is installed, verify that the following requirements have been met:

SAP System
: The SAP Kernel Release 4.6C or higher is required.

  The SAS Federation Server Driver for SAP for Windows® and UNIX® requires the 64-bit SAP Unicode RFC library, Release 7.10, which is provided by SAP AG. Refer to SAP Note 413708 for the current version, download and installation instructions.

SAPGUI
: During the installation of the SAP components, a SAPGUI is required.

User IDs
: An SAP user ID and password is required. The user ID must have appropriate authorizations to access data and use communication methods. For more information about customizing the authorization, see Authorization Profiles below.

To install and run SAS Federation Server Driver for SAP, the following SAP user IDs are required:

- **RFC user** This is an SAP user ID that is used for the communication link between the SAS Federation Server Driver for SAP and the SAP System Application Server. Typically, there are several RFC user IDs (one per person).
- **SAP System Administrator** An SAP System Administrator ID is required for the installation of ABAP programs and function modules, for the configuration of destinations and variant for batch operations, and for setting up authorizations for user IDs to use the SAS Federation Server Driver for SAP. This user ID is used only for the installation.

Connectivity

The SAS Federation Server Driver for SAP and the SAP Application Server usually use TCP/IP communication. Refer to the RFC documentation from SAP AG. The host of the SAP Application Server must be known by the host of the SAS Federation Server Driver for SAP. Alternatively, you can use the IP address to identify the SAP System Application Server. The TCP/IP services file must contain entries for the services, ports, and protocols used for the communication.

The following is an example for entries in the services file:

```
sapdp00 3200/tcp
sapdp01 3201/tcp
sapdp99 3299/tcp
sapgw00 3300/tcp
sapgw01 3301/tcp
...
sapgw99 3399/tcp
sapsp00 3400/tcp
sapsp01 3401/tcp
...
sapsp99 3499/tcp
```

Note: If the SAPGUI is installed on the system, the TCP/IP services file already contains these entries.

Authorization Profiles

To install and use the SAS Federation Server Driver for SAP, a user ID with certain authorizations is required. An authorization has an authorization object. Several authorizations can be bundled together into an authorization profile.

If the batch functionality of the SAS Federation Server Driver for SAP is used, the RFC user ID needs to have authorization to submit batch jobs already released.

The RFC user IDs require authorizations for the following authorization objects:

Object	Minimum Requirement for Values	Example for Predefined Authorization
S_RFC (Authorization check for RFC access)	ACTVT: * RFC_NAME: * RFC_TYPE: *	S_RFC_ALL
S_TABU_DIS (Table maintenance via standard tools such as SM31)	ACTVT: 03 DICBERCLS: *	S_TABU_SHOW
S_BTCH_JOB (Background processing: Operations on Background Jobs) [1]	JOBACTION: RELE JOBGROUP: *	

[1] Required only if batch functionality of the RFC server is used.

The existing authorizations, for example S_TABU_SHOW, can be used. The S_RFC and the S_TABU_DIS authorizations are in authorization profile A_ANZEIGE.

Setting up the SAS Federation Server Driver for SAP

This section describes the set up for the SAS Federation Server Driver for SAP after the software has been installed.

Complete the following steps on SAS Federation Server:

1. Install the Unicode RFC libraries from SAP.

   The SAS Federation Server Driver for SAP requires the 64-bit version of the SAP Unicode RFC libraries, Release 7.10. The libraries must be installed on SAS Federation Server. These shared libraries are provided by SAP AG. Download and install the 64-bit Unicode libraries for your platform, following the instructions in SAP Note 413708.

2. Set the environment variables.

   The SAS Federation Server Driver for SAP executable uses the SAP shared libraries. You must add the location of the SAP RFC shared libraries to the shared library path environment variable specific to your operating system. For Windows, ensure that the shared libraries are installed in the system path, or add the directory of the installed SAP Unicode RFC libraries to the Path environment variable. For UNIX, replace rfclib_directory in the table below with the directory where the RFC shared libraries are installed.

Table 7.1 *AIX*

Bourne Shell	`$ LIBPATH=rfclib_directory:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH` `rfclib_directory:$LIBPATH`

Table 7.2 *HP-UX*

Bourne Shell	`$`
	`LD_LIBRARY_PATH=rfclib_directory:$LD_LIBRARY_PATH`
	`$ export LD_LIBRARY_PATH`

C Shell	`$ setenv` `LD_LIBRARY_PATH=rfclib_directory:$LD_LIBRARY_PATH`

Table 7.3 *HP-UX for the Itanium Processor Family Architecture*

Bourne Shell	`$`
	`LD_LIBRARY_PATH=rfclib_directory:$LD_LIBRARY_PATH`
	`$ export LD_LIBRARY_PATH`

C Shell	`$ setenv` `LD_LIBRARY_PATH=rfclib_directory:$LD_LIBRARY_PATH`

Table 7.4 Linux for Intel Architecture, Linux for x64, Solaris, and Solaris for x64

Bourne Shell	`$ LD_LIBRARY_PATH=rfclib_directory:$LD_LIBRARY_PATH` `$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH=rfclib_directory:$LD_LIBRARY_PATH`

Installing SAP Components

Verify the Prerequisites

Make sure that you have fulfilled the following prerequisites:

SAPGUI Prerequisites
: The installation of the SAS Federation Server Driver for SAP components require SAPGUI software to be installed on your PC or workstation.

Note: Although it is not absolutely necessary to have the SAPGUI installed on the same PC or workstation where the SAS Federation Server Driver for SAP is going to be installed, you need access to the SAPGUI during the installation. Because the usage of the SAPGUI complements SAP functionality, it is recommended that the SAPGUI be installed on the same PC or workstation.

SAP Administrator ID Prerequisites
: A valid SAP user ID and password is required. The user must have authorization to transport files and for RFC destination maintenance. It is strongly recommended to get assistance from your SAP System Administrator to perform these tasks.

Install ABAP Programs and Function Modules

Delivery transport files are included in the SAS Federation Server Driver for SAP. These transport files include all of the components, ABAP programs and function modules needed to run the SAS Federation Server Driver for SAP.

The delivery transports have to be imported on each SAP application server that is going to be accessed by SAS Federation Server. If an SAP system is upgraded, the delivery transports have to be imported again.

Two sets of transports are included, one for releases prior to SAP Release 7.0 and one for SAP Release 7.0 and above. You must import the transport files that apply to your system.

Version	Transport	Purpose	To be applied to
SAP systems prior to SAP NetWeaver 7.0 (Kernel 6.40 or earlier)	SAPKA93120INSAS *Note:* This transport must be installed first.	Supports the SAS Federation Server Driver for SAP	All SAP systems to be accessed by the SAS Federation Server Driver for SAP
SAP NetWeaver 7.0 based systems and later	SAPKA93130INSAS *Note:* This transport must be installed first.	Supports the SAS Federation Server Driver for SAP	All SAP systems to be accessed by the SAS Federation Server Driver for SAP

Version	Transport	Purpose	To be applied to
	SAPKB93031INSAS	Supports new BI 7.0 authorization concept	Optional; SAP BI 7.0 systems and above; only apply if you are using the new authorization concept

To import the transport files to your SAP systems, follow the instructions below. The instructions are based on the usage of the tp program (a utility for transport between SAP systems) on the operating system level.

Note: Replace HOME in these instructions with the actual directory path where SAS Federation Server is installed.

1. Log in as SAP System Administrator to the SAP application server.
2. Move the transport files from SAS Federation Server into the appropriate directories on your SAP systems.

Windows	Copy the r3trans.exe file to your SAP application server and extract the files into the transport directory, for example, HOME:\share\SAP. The files for all transports will be put into the cofiles and data subdirectories.
UNIX	Copy the r3trans.tar file to your SAP application server and extract the files into the transport directory, for example, HOME:/share/SAP. Assuming that the tar file is downloaded to the user's HOME directory, follow these procedures to extract the files into the cofiles and data subdirectory in **/usr/sap/trans**. .**$ cd /usr/sap/trans$ tar -xvf $HOME/r3trans.tar**

3. Change to the transport program directory using the following command:

Windows	**<drive>:** **cd \usr\sap\trans\bin**
UNIX	**$ cd /usr/sap/trans/bin**

4. Load the transport into the transport buffer and import the transport into your SAP system with the following commands. Replace *SID* with the system ID for your SAP system.

```
tp addtobuffer  SAPKA93120INSAS SID
tp import SAPKA93120INSAS SID U2
```

Note:

1. Make sure you are using the correct profile for the transport control program tp. In some cases it might be necessary to use the parameter pf= to specify the TPPARAM file.
2. Because the transport file uses a long name, the nbufform=true TP option must be set. The option can either be maintained in the SAP system using transaction STMS, or it can be specified as a parameter to the tp command. Also, the TP option tp_version= must be set to at least 264 to allow the long names.

3. The U2 option allows the originals to be overwritten if the user has previously installed these ABAP objects.
4. The transports contain only client-independent ABAP objects. The tp import can therefore use any existing client that is correctly set up for imports. Verify that the ABAP program RDDIMPDP is correctly scheduled in the client that you use for the import.
5. If the transport files are imported into a Unicode SAP system, use the transport profile parameter "setunicodeflag=true" to force setting the Unicode flags in the imported programs. Refer to SAP Note 330267 for more details. The "setunicodeflag=true" is not necessary if you are using the transports for SAP NetWeaver 7.0 based system and higher. Those transports have been created with the Unicode flag.

Considering these notes, the tp commands might require additional parameters. Replace *SID* with the system ID for the SAP system.

Note: The tp commands listed on several lines in the following examples should be entered on a single command line. Be sure to include a space before adding the text from each of the following lines.

SAP Release prior to SAP NetWeaver 7.0 (Kernel 6.40 or lower), non-Unicode SAP Server

Windows

```
tp addtobuffer
SAPKA93120INSASSID
pf=\usr\sap\trans\bin\TP_DOMAIN_SID.PFL
-D"nbufform=true" -D"tp_version=264"

tp import
SAPKA93120INSASSID
pf=\usr\sap\trans\bin\TP_DOMAIN_SID.PFL
-D"nbufform=true" -D"tp_version=264"
```

UNIX

```
$ tp addtobuffer
SAPKA93120INSAS SID
pf=/usr/sap/trans/bin/TP_DOMAIN_sid.PFL
-D"nbufform=true" -D"tp_version=264"

$ tp import
SAPKA93120INSASSID
pf=/usr/sap/trans/bin/TP_DOMAIN_sid.PFL
-D"nbufform=true" -D"tp_version=264"
```

SAP Release prior to SAP NetWeaver 7.0 (Kernel 6.40 or lower), Unicode SAP Server

Windows

```
tp addtobuffer
SAPKA93120INSASSID
pf=\usr\sap\trans\bin\TP_DOMAIN_SID.PFL
-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"

tp import
SAPKA93120INSASSID
pf=\usr\sap\trans\bin\TP_DOMAIN_SID.PFL
-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"
```

UNIX	`$ tp addtobuffer`
	`SAPKA93120INSAS`*`SID`*
	`pf=/usr/sap/trans/bin/TP_DOMAIN_`*`SID`*`.PFL`
	`-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"`
	
	`$ tp import`
	`SAPKA93120INSAS`*`SID`*
	`pf=/usr/sap/trans/bin/TP_DOMAIN_`*`SID`*`.PFL`
	`-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"`

SAP NetWeaver 7.0 based systems and later, Unicode SAP Server	

Windows	`tp addtobuffer`
	`SAPKA93130INSAS`*`SID`*
	`pf=\usr\sap\trans\bin\TP_DOMAIN_`*`SID`*`.PFL`
	`-D"nbufform=true"`
	
	`tp import`
	`SAPKA93130INSAS`*`SID`*
	`pf=\usr\sap\trans\bin\TP_DOMAIN_`*`SID`*`.PFL`

UNIX	`$ tp addtobuffer`
	`SAPKA93130INSAS`*`SID`*
	`pf=/usr/sap/trans/bin/TP_DOMAIN_`*`sid`*`.PFL`
	`-D"nbufform=true"`
	
	`$ tp import`
	`SAPKA93130INSAS`*`SID`*
	`pf=/usr/sap/trans/bin/TP_DOMAIN_`*`SID`*`.PFL`
	`-D"nbufform=true"`

Maintaining RFC Destinations

Note: If the SAS Federation Server Driver for SAP will execute requests using the SAP batch processing facility (recommended), you must complete this section.

The SAS Federation Server Driver for SAP uses multiple RFC destinations (TCP/IP connection type) for accessing an SAP System in batch. The number of destinations setup for the SAS Federation Server Driver for SAP limits the number of concurrent requests to the SAP application server.

For example, create six destinations with connection type T and activation type Registered Server Program that can be used by the SAS® server. The program ID for the registered server program must be unique on the SAP gateway.

RFC Destination Name	Program ID
SAS1	RFC.SAS1
SAS2	RFC.SAS2
SAS3	RFC.SAS3
SAS4	RFC.SAS4

RFC Destination Name	Program ID
SAS5	RFC.SAS5
SAS6	RFC.SAS6

Complete the following steps:

1. Call transaction SM59 in SAP. Specify transaction code /nsm59 in the command field.
2. Click **Create**.
3. Enter SAS1 as the RFC destination.
4. Enter T as the Connection type.
5. Enter a description for the destination.
6. Click **Enter**.
7. Choose **Registration for the Activation Type** or **Registered Server Program** in the **Technical Settings** tab.
8. Enter the RFC.SAS1 as the program ID.
9. If required, enter the gateway host and gateway service in the Gateway Options panel. The gateway host is the host name of the local gateway and gateway service is usually sapgwsysnr, where sysnr is replaced by the system number of the SAP system.
10. (Unicode SAP systems only) Select the Unicode on the **MDMP & Unicode** tab. Ignore the message about performing the Unicode test. The Unicode test cannot be performed with the destinations created for the SAS Federation Server Driver for SAP.
11. Save the destination.
12. Repeat step 1 through 11 for each of the new RFC destinations.

Maintaining the /SAS/DESTS Table

The RFC destinations defined in the previous step must be grouped into destination groups. The groups are defined in table /SAS/DESTS which is used for controlling the access to the destinations from the SAS Federation Server that accesses the SAP system.The destination group is a parameter of the SAS Federation Server Driver for SAP. The default is "SAS1".

Complete the following steps:

1. Call transaction SM30 in SAP. On the command line, type transaction code **`/nsm30`**.
2. In the **Table** field, enter the table name /SAS/DESTS.
3. In the **Restrict Data Range** field, select **No Restrictions**.
4. Click **Maintain**.
5. An information message appears. Click **OK**.
6. Click **New Entries**.
7. For each of the RFC destinations that you defined in step 2, enter the destination group ID as the SAS ID and the RFC destination name. The following examples define the destinations for destination group SAS1:

SAS ID	RFC Destination	Used
SAS1	SAS1	
SAS1	SAS2	
SAS1	SAS3	
SAS1	SAS4	
SAS1	SAS5	
SAS1	SAS6	

8. Save the table.

Activating BAdI Implementation

The SAS Federation Server Driver for SAP has three basic implementations for table access authorization checks. The default implementation uses the SAP authorization object S_TABU_DIS to check the authorization. If you want to use any of the other two implementations you have to activate the appropriate BAdI implementation.

Table 7.5 *BAdI Implementation*

Default	Authorization object S_TABU_DIS
Classic BAdI /SAS/AUTHBW01	For BW and BI: User authorization checks at the InfoCube, InfoObject and ODS level using the reporting authorization (SAP standard authorization concept).
New BAdI enhancement / SAS/IM_AUTHBI01	For BI 7.0+ only: User authorization checks using the analysis authorization. This not only provides an authorization check for the infoProvider (infoCube, infoObject, DSO) but also column level restrictions on master data attributes and key figures, and row-level restrictions on attributes.

In releases prior to SAP NW BI 7.0, SAP uses the reporting authorization concept that uses the SAP standard authorization concept. If you want to activate the SAS implementation for those authorization checks:

1. Call transaction SE19 in SAP. In the command field, type transaction code **`/nse19`**.
2. Enter **`/SAS/AUTHBW01`** as the implementation.
3. Click **Activate**.

In BI 7.0, SAP introduces a new authorization concept for analysis authorization. If you want to use the SAS implementation for those authorization checks, import the appropriate transport (SAPKB92331INSAS). The implementation is activated by default. If you want to deactivate the implementation:

1. Call transaction SE19 in SAP. In the command field, type transaction code **`/nse19`**.

2. In the **Edit Implementation** field, select the **New BAdI** check box.
3. Enter **`/SAS/IM_AUTHBI01`** as the enhancement implementation.
4. Click **Change**.
5. Double-click the **BAdI Implementation** to deactivate (such as / SAS/BADI_CHECK_FILTER) and clear the **Implementation is active** check box in the **Runtime Behavior** field. Repeat for each of the implementations listed in the left hand side of the **Enh.Implementation Elements** tab.
6. Save and activate the changes.

SAS Data Set Reference

Overview

The SAS data set is a SASProprietary file format, which contains data values organized as a table of rows (SAS observations) and columns (SAS variables). The supported file format is the same as a SAS data set that is created by the BASE engine in SAS for Version 7 and later. A supported SAS data set uses the extension .**`sas7bdat`**.

Understanding the Driver for Base SAS

The SAS Federation Server Driver for Base SAS is a SASProprietary driver that provides Read and Update access to legacy SAS data sets and creates SAS data sets that can be accessed by both the legacy and SAS Federation Server data access services.

The driver supports much of the Base SAS functionality, such as SAS indexing and general integrity constraints, as well as much of the Federation Server SQL (FedSQL) functionality.

The SAS Federation Server Driver for Base SAS is an in-process driver, which means that it accesses data in the same process that executes the data access services. All server connections made with the SAS Federation Server Driver for Base SAS use LOCKTABLE=SHARED and PATH_BIND=ACCESS.

Connection Options for SAS Data Sets

Connection Options

To access data that is hosted on SAS Federation Server, a client must submit a connection string, which defines how to connect to the data. The data service connection arguments for a SAS data set include connection options and advanced options.

The following connection options are supported for SAS data sets:

Option	Description
CATALOG	**`CATALOG=`*`catalog-identifier`*`;`** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid (for example, catalog=base). A catalog name can be up to 32 characters long. *Note:* You must specify a catalog.

Option	Description
DRIVER	**DRIVER=BASE;** Identifies the data service to which you want to connect, which is a SAS data set. *Note:* You must specify DRIVER=BASE to access a SAS data set.
(SCHEMA) NAME	**NAME=*schema-identifier*;** Specifies an arbitrary identifier for an SQL schema. Any identifier is valid (for example, name=myfiles). The schema identifier is an alias for the physical location of the SAS library, which is much like the Base SAS libref. A schema name must be a valid SAS name and can be up to 32 characters long. *Note:* You must specify a schema identifier.
PRIMARY PATH	**PRIMARYPATH=*physical-location*;** Specifies the physical location for the SAS library, which is a collection of one or more SAS files. For example, in directory-based operating environments, a SAS library is a group of SAS files that are stored in the same directory. *Note:* You must specify a primary path.
SCHEMA ATTRIBUTES	**SCHEMA=(*attributes*);** Specifies schema attributes that are specific to a SAS data set. A schema is a data container object that groups tables. The schema contains a name, which is unique within the catalog that qualifies table names. For a SAS data set, a schema is similar to a SAS library, which is a collection of tables and which has assigned attributes.

Advanced Options

Advanced driver options are additional options that are not required in order to connect to the data source. They are used to establish connections to catalogs, data source names (DSNs), and schemas. Although advanced options can also be used when connecting to a data service, doing so will cause the specified options to apply to all data service connections.

The following advanced options are supported for SAS data sets:

Option	Description
ACCESS	**ACCESS=READONLY | TEMP;** • **READONLY** Assigns a read-only attribute to the schema. You cannot open a SAS data set to update or write new information. • **TEMP** specifies that the SAS data sets be treated as scratch files. That is, the system will not consume CPU cycles to ensure that the files do not become corrupted. **TIP** Use ACCESS=TEMP to save resources only when the data is recoverable. If TEMP is specified, data in memory might not be written to disk on a regular basis. This saves I/O, but could cause a loss of data if there is a crash.

Option	Description
CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.
COMPRESS	**COMPRESS=NO | YES | CHAR | BINARY;** Controls the compression of rows in created SAS data sets. • **NO** Specifies that the rows in a newly created SAS data set are uncompressed (fixed-length records). This setting is the default. • **YES | CHAR** Specifies that the rows in a newly created SAS data set are compressed (variable-length records) by using RLE (Run Length Encoding). RLE compresses rows by reducing repeated consecutive characters (including blanks) to two- or three-byte representations. *TIP* Use this compression algorithm for character data. • **BINARY** Specifies that the rows in a newly created SAS data set are compressed (variable-length records) by using RDC (Ross Data Compression). RDC combines run-length encoding and sliding-window compression to compress the file. *TIP* This method is highly effective for compressing medium to large (several hundred bytes or larger) blocks of binary data (numeric columns). Because the compression function operates on a single record at a time, the record length must be several hundred bytes or larger for effective compression.

Option	Description
DEFAULT_ATTR	**DEFAULT_ATTR=(*attr=value;...*)** DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **attr=value** corresponds to any of the options below: • **CURSORS=*n*** - Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **DEFAULT_ATTR=(CURSORS=2)** • **USE_EVP=*n***— Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **DEFAULT_ATTR=(USE_EVP=0)** • **XCODE_WARN=*n*** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **DEFAULT_ATTR=(XCODE_WARN=1)**
ENCODING	**ENCODING=*encoding-value;*** Overrides and transcodes the encoding for input or output processing of SAS data sets. *Note:* The default value is the current operating system setting.
LOCKTABLE	**LOCKTABLE=SHARED | EXCLUSIVE** Places exclusive or shared locks on SAS data sets. You can lock tables only if you are the owner or have been granted the necessary privilege. The default value is SHARED. • **SHARED** Locks tables in shared mode, allowing other users or processes to read data from the tables, but preventing other users from updating. • **EXCLUSIVE** Locks tables exclusively, preventing other users from accessing any table that you open.

Teradata Reference

Understanding the SAS Federation Server Driver for Teradata

The SAS Federation Server Driver for Teradata provides Read and Update access to Teradata database tables and creates tables that can be accessed by both SAS Federation Server and Teradata.

The SAS Federation Server Driver for Teradata supports most of the FedSQL functionality. The driver also supports an application's ability to submit native Teradata SQL statements.

The SAS Federation Server Driver for Teradata is a remote driver, which means that it connects to a server process to access data. The process might be running on the same machine as SAS Federation Server, or it might be running on another machine in the network.

Connection Options for Teradata

Overview

To access data that is hosted on SAS Federation Server, a client must submit a connection string, which defines how to connect to the data. The data service connection arguments for a Teradata database include connection options and advanced options.

Connection Options

Connection options are used to establish a connection to a data source. Specify one or more connection options when defining a data service using the CREATE DATA SERVICE DDL statement on page 163 .

The following connection options are supported for a Teradata database.

Option	Description
CATALOG	**CATALOG=*catalog-identifier*;** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid (for example, catalog=tera). *Note:* You must specify a catalog.
DATABASE	**DATABASE=*database-name*;** Specifies the Teradata database. If you do not specify DATABASE=, you connect to the default Teradata database, which is often named the same as your user ID. If the database value that you specify contains spaces or non-alphanumeric characters, you must enclose it in quotation marks.
DRIVER	**DRIVER=TERA;** Identifies the data service to which you want to connect, which is a Teradata database. *Note:* You must specify the driver.
SERVER	**SERVER=*server-name*;** Specifies the Teradata server identifier.

Advanced Options

The following advanced options are supported for Teradata database.

Option	Description
ACCOUNT	**ACCOUNT=*account-ID*;** Specifies an optional account number that you want to charge for the Teradata session.

Option	Description
CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping will be disabled when CT_PRESERVE is set to STRICT. If the requested type does not exist on the target database, an error is returned. The options are: • **STRICT** The requested type must exist in the target database. No type promotion will occur. If the type does not exist, an error is returned. • **SAFE** Target data types will be upscaled only if they will not result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE** This is the default for all drivers. The best corresponding target data type will be chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size will be recalculated to ensure all characters can be stored in the new encoding. • **FORCE_COL_SIZE** This option is the same as FORCE, except that the column size for the new encoding will be the same as the original encoding. This can be used to avoid column size creep that has been seen in some cases, but it means that the resulting column might be too large or too small for the target data.
DEFAULT_ATTR	**DEFAULT_ATTR=(*attr=value;...*)** DEFAULT_ATTR is used to specify connection handle or statement handle attributes supported for initial connect-time configuration. Where **attr=value** corresponds to any of the options below: • **CURSORS=*n*-** Connection handle option. This option controls the driver's use of client side result set cursors. The possible values are 0, 1 or 2. 0 Causes the driver to use client side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one. 1 Causes the driver to always use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will never be used. 2 (Default) Causes the driver to never use client side static cursor emulation if a scrollable cursor is requested. The database server's native cursor will be used if available – otherwise the cursor will be forward only. Example: **DEFAULT_ATTR=(CURSORS=2)** • **USE_EVP=*n*—** Statement handle option. This option optimizes the driver for large result sets. The possible values are 0 (OFF) or 1 (ON), which is the default. Example: **DEFAULT_ATTR=(USE_EVP=0)** • **XCODE_WARN=*n*** - Statement handle option. Used to warn on character transcoding errors that occur during row input or output operations. Possible values are 0 (returns an error), 1 (returns a warning), or 2 (ignore transaction errors). Example: **DEFAULT_ATTR=(XCODE_WARN=1)**

Option	Description
DRIVER TRACE	**DRIVER_TRACE='API | SQL | ALL';** Requests tracing information, which logs transaction records to an external file that can be used for debugging purposes. The SAS Federation Server driver writes a record of each command that is sent to the database to the trace log based on the specified tracing level, which determines the type of tracing information. The tracing levels are: • **ALL** Activates all trace levels. • **API** Specifies that API method calls be sent to the trace log. This option is most useful if you are having a problem and need to send a trace log to Technical Support for troubleshooting. • **DRIVER** Specifies that driver-specific information be sent to the trace log. • **SQL** Specifies that SQL statements that are sent to the database management system (DBMS) be sent to the trace log. Tracing information is DBMS specific, but most SAS Federation Server drivers log SQL statements such as SELECT and COMMIT. **Default:** Tracing is not activated. *Note:* If you activate tracing, you must also specify the location of the trace log with DRIVER_TRACEFILE=. Note that DRIVER_TRACEFILE= is resolved against the TRACEFILEPATH set in ALTER SERVER. TRACEFILEPATH is relative to the server's Content Root location. (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=. **Interaction:** You can specify one trace level, or you can concatenate more than one by including the | (OR) symbol. For example: **driver_trace='api|sql'** generates tracing information for API calls and SQL statements.
DRIVER TRACE FILE	**DRIVER_TRACEFILE='*filename*';** Used to specify the name of the text file for the trace log. Include the filename and extension in single or double quotation marks. For example: **driver_tracefile='\mytrace.log'** **Default:** The default TRACEFILE location applies to a relative filename, and it will be placed relative to TRACEFILEPATH. **Requirement:** DRIVER_TRACEFILE is required when activating tracing using DRIVER_TRACE. **Interaction:** (Optional) You can control trace log formatting with DRIVER_TRACEOPTIONS=.
DRIVER TRACE OPTIONS	**DRIVER_TRACEOPTIONS=APPEND | THREADSTAMP | TIMESTAMP;** Specifies options in order to control formatting and other properties for the trace log: • **APPEND** Adds trace information to the end of an existing trace log. The contents of the file are not overwritten. • **THREADSTAMP** Prepends each line of the trace log with a thread identification. • **TIMESTAMP** Prepends each line of the trace log with a time stamp. **Default:** The trace log is overwritten with no thread identification or time stamp.
PASSWORD	**PASSWORD=*password*;** Specifies a Teradata password. The password must correlate to your USER= value. The alias is PWD=. *Note:* You must specify the PASSWORD= option.
ROLE	**ROLE=*security-role*;** Specifies a security role for the session.

Option	Description
USER	**USER=*user-id*;** Specifies a Teradata user ID. If the ID contains blanks or national characters, enclose it in quotation marks. The alias is UID=. *Note:* You must specify the USER= option.

Appendix 1

Administration DDL Statements Reference

ALTER SERVER Statement

Enables you to change the server configuration by specifying server options. Following is the syntax:

```
ALTER SERVER

    [ alter-server-options ]
```

Following are the options for the ALTER SERVER statement:

alter-server-options
Specifies the list of server options to alter. Shared login manager and password options are paired.

```
alter-server-options ::=

      "{" OPTIONS ["("] alter-server-option

                [{ "," alter-server-option } ... ] [")"]"}"
```

alter-server-option
Specifies the server option to alter.

```
alter-server-option ::=

      [alter-operation ]server-option | cache-option
```

server-option
Specifies the server configuration. One of the following:

FILEDSNPATH | FILEDSN_ROOT *directory-path*
File DSN content location root for FILEDSN and SAVEFILE keywords. This path can be absolute or relative to the server's root content location specified in the server configuration file.

PURGE_CACHE *time-out-value*
Specifies how often (in minutes) old data cache tables will be removed from the server.

- A positive time value indicates the interval at which the cleanup thread will check for cleanup.
- A negative time-out-value means cleanup will happen in response to an explicit PURGE CACHE command only.
- A value of 0 indicates that old caches will be removed after a CREATE, REFRESH, or PURGE CACHE command is issued.

TRACEFILEPATH | TRACEFILE_ROOT *directory-path*
Specifies the trace file content location root for TRACEFILE keywords and the TRACEFILE environment handle attribute. The trace file path is used to store all trace files that are created, either from SAS Federation Server start-up, or when enabled on a connection. This path may be absolute or relative to the server's ContentRoot that is specified in the server configuration file. The default trace file path is **drive:\Program Files\SASHome\FedServer\server instance \var**.

The DRIVER_TRACEFILE= path that is set in the connection option, DRIVER TRACE, is resolved against the path that is set here.

SHAREDLOGINMANAGER *manager*
The Authentication Server manager account that will be used to retrieve the shared login map credentials.

SHAREDLOGINPASSWORD *password*
The Authentication Server password for the manager account that will be used to retrieve the shared login map credentials.

Note: Shared login manager and password options are paired.

SHAREDLOGINKEY *key*
The Authentication Server grouping key used to search for shared login map credentials.

TRUSTED_USER_UID *userid*
The Authentication Server trusted user account that is used to act on behalf of other users for purposes of retrieving user-owned logins and group memberships (trusting that those users have already been authenticated). The SAS Federation Server makes a trusted user connection to Authentication Server to process queries on definer's rights views since these views retrieve SQL data under the identity of the view's schema owner (the definer) rather than the invoking user.

TRUSTED_USER_PWD *password*
Specifies the password for the TRUSTED_USER_UID.

Note: The Shared Login and Trusted User passwords are encrypted. By default, SASProprietary encryption is used and the method of encryption can vary depending on the security platform where SAS Federation Server is installed. If DataFlux Secure for SAS Federation Server has been installed, AES encryption can be used.

CONNECTION_POOLING [N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1]
This option controls whether connection pooling is enabled or disabled for the server. If connection pooling is switched on, connections to databases are not disconnected immediately when the client requests to disconnect from the database. The connections are put into a pool of connections that can be reused by subsequent requests to connect to the same database with the same attributes and credentials

CONNECTION_POOL_TIMEOUT *seconds*
This option identifies the time in seconds an unused connection stays in the connection pool. The default is 60 seconds. If the option is unset or set to 0, the connection stays in the pool for 60 seconds. If the time is exceeded the connection is removed from the pool and the connection is closed.

CONNECTION_POOL_MAXSIZE *maxsize*
This option identifies the maximum number of unused connections in the connection pool. The default is 50. If the option is unset or set to 0, a maximum number of 50 connections are kept in the pool. If the maximum number of connections is reached and a new connection is added to the connection pool, the oldest connection is removed from the pool and the connection is closed.

cache-option : :=
Specifies the server configuration. One of the following:

CACHE (NAME *cache-name, cache-property cache-property value*)
The NAME option specifies the cache name. Cache properties for that particular cache are altered or created within the sublist. Normal generic SQL options syntax and rules apply to the CACHE option and its suboptions.

cache name : :=
Specifies the name of the Authentication Server cache.

```
AS - All AS cached resources

AS.Name - All AS.Name mappings

AS.Name.Subjects - User name to AS identifier cache

AS.Name.Groups - Group name to AS identifier cache

AS.Subject - All AS.Subject cache resources

AS.Subject.Groups - User group memberships cache
```

```
AS.Subject.Principals - User principal listings cache

AS.List - Listings

AS.List.Subjects - User listings cache

AS.List.Groups - Group listings cache

Authorization - Privileges

ResultSet - Result set caches

ResultSet.View - Materialized view cache (VDC)
```

cache-property : :=
A property of the cache.

TIMEOUT
TIMEOUT is the number of seconds before the cache data becomes stale after a refresh. After timing out, the cache is emptied and refreshed on demand or emptied automatically, depending on the cache implementation. The timeout can be set for multiple related caches by specifying a non-terminal cache namespace for the name suboption such as **`AS.List`** instead of **`AS.List.Groups`**. A value of -1 corresponds to infinite and a value of 0 corresponds to immediate. The default time-out value is 0 for all caches if unset directly or through a parent namespace.

SQL Statement Limit

To control the amount of memory available to answer SQL requests, and enforce this limit for all connections, use the CONOPTS option with the FedSQL driver and set the SQL statement limit as explained below. If the option is specifically set on a particular DSN, then the DSN value should override the system setting.

```
ALTER SERVER {options CONOPTS(driver FEDSQL,
xset DEFAULT_ATTR(SQL_STMT_MEM_LIMIT 8000000))}
```

CONOPTS
Use this connection string option to call the FedSQL driver.

Driver FEDSQL
FedSQL is the required driver for SQL requests on SAS Federation Server.

DEFAULT_ATTR(SQL_STMT_MEM_LIMIT = *n*)
FedSQL statement handle option. Used to control the amount of memory available to FedSQL to answer SQL requests. (*n*)umber is treated as an integer and is specified in bytes.

Examples:

```
ALTER SERVER {OPTIONS add TRACEFILEPATH "C:\tracefiles"}

ALTER SERVER {OPTIONS add TRACEFILEPATH "logs\tracefiles"}

ALTER SERVER {OPTIONS xset SHAREDLOGINKEY 'DefaultKey'}

ALTER SERVER {OPTIONS( CACHE(NAME AS.Subject, TIMEOUT 300),
 CACHE(NAME AS.List.Subjects, TIMEOUT 60) )}

ALTER SERVER {OPTIONS(xset PURGE_CACHE 30)}

ALTER SERVER {options CONOPTS(driver FEDSQL,
xset DEFAULT_ATTR(SQL_STMT_MEM_LIMIT 8000000))}
```

CREATE DATA SERVICE Statement

The CREATE DATA SERVICE statement enables you to create a data service using the following syntax:

```
CREATE [DATA] SERVICE data-service

TYPE data-service-type

[CATALOG [NAME] catalog-name]

[DOMAIN [NAME] domain-name]

[REGISTER [( catalog-name1 [,catalog-name2 …]) | ALL]

[register-options]]

[data-service-options]
```

CREATE [DATA] SERVICE *data-service*
Specifies the data service name. The following rules apply:

- The specified name must not match the reserved name of the internal data service, BASE.
- The specified name must not match the name of any existing data service.
- The specified name must not match the name of an existing catalog for defined data services that do not support catalog names, unless the CATALOG option is used to specify a different name.

TYPE *data-service-type*
Specifies the data service type. One of:

```
DB2UNXPC

GENERIC

GENERIC_FED

GREENPLUM

ODBC

ODBC_FED

ORACLE

MDS

SQLSVR

TERADATA
```

CATALOG [NAME] *catalog-name*
Specifies the logical catalog name associated with the data service. The catalog name must be unique. If an identical catalog name is encountered, a warning message is issued. The default logical catalog name matches the data service name if omitted for data sources that do not support catalogs.

DOMAIN [NAME] *domain-name*
Specifies the Authentication Server domain name associated with this data service. If omitted, the default domain name matches the data service name.

REGISTER [(*catalog-name1* [, *catalog-name2*...]) | *ALL*]
Catalog registration specification. (Optional) Specify a list of catalogs to register or ALL keyword to register all catalogs visible to the connection.

```
register-options  : :=
    [ UID "userid" ]
    [ PWD 'password' ]
    [ VALIDATE [N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1]]
```

Note: When using register options, the UID requires double quotation marks and the PWD requires single quotation marks.

UID "*userid*"
Data service user ID.

PWD '*password*'
Data service user password.

VALIDATE [N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1]
Specifies whether the connection and the catalog names are to be validated.

VALIDATE defaults to TRUE if not specified or if no Boolean value keyword is specified and either UID or PWD is specified. If VALIDATE is true and neither UID, nor PWD are specified, then a user ID and password (personal credentials) are extracted on behalf of the caller from the domain associated with the data service. The statement returns TKTS_ERROR if no such credentials exist.

If an explicit catalog list is specified and VALIDATE is TRUE, then each catalog must exist or the statement returns TKTS_ERROR. If VALIDATE is TRUE but no catalogs are specified (including REGISTER ALL), then only the connection itself is validated since the catalog list is either empty or supplied by the connection.If an attempt is made to register a catalog that has already been registered or associated with a data service, then the statement returns TKTS_SUCCESS_WITH_INFO.

data-service-options
Specifies the list of data service options.

```
data-service-options ::=

       "{" OPTIONS ["("") data-service-option

                       [{ "," data-service-option } ... ] [")"]"}"
```

data-service-option
Specifies the data service option.

```
data-service-option ::=

data-service-dependent-option |

data-service-independent-option
```

data-service-dependent-option
Data service specific options. For details about the specific options, see the Driver Reference chapter.

GENERIC
The following options apply for GENERIC data services:

```
LOCAL N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1
```

Specifies if the data service refers to a local data source or an external database. Local sources do not require secondary authentication; and as such, specification of

the DOMAIN clause results in an error if the LOCAL option is specified as one of the true values. The default is NO. This option is not persisted.

GENERIC_FED

Any options that apply to multi-catalog data services will apply to GENERIC_FED data services.

```
GENERIC_FED-option ::= GENERIC-option
```

data service independent option

```
data-service-independent-option ::=

conopts-configuration-list | case-sensitivity-option

conopts-configuration-list ::=

CONOPTS "(" [DRIVER driver-name] ","
driver-connection-string-option ...] ")" ...
```

driver-name

If the **`driver-name`** option is omitted, the default driver for the data service is assumed. Associated options within the CONOPTS list are used for connections using the appropriate driver. Some data services such as ORACLE accept connections from the ODBC driver as well. For these data services, two CONOPTS lists can be configured, one per driver to accept connections for the two drivers. The ODBC driver accepts a CONOPTS driver connection string option. To configure this option and suboptions within it, the configuration option format would be CONOPTS(driver ODBC, conopts(...)). The inner CONOPTS option happens to be a specific driver connection string (list-valued) option while the outer CONOPTS groups arbitrary driver connection string options configured for the service.

driver-connection-string-option

Specifies the connection options that correspond to the driver which is specified in DRIVER driver-name. The DATA_SERVICE and CATALOG connection string options should not be specified here since they are implied by the data service and its configured catalogs.

```
case-sensitivity-option ::=
CASE_SENSITIVITY "("
OBJECT N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1 ","
COLUMN N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1 ")"
```

This option specifies the case sensitivity to use when comparing identifiers for security purposes. False indicates case insensitive compares while True indicates case sensitive compares will be used. If not specified, the value for case sensitivity will be set to the default for the data service. (True/Sensitive for DB2, Oracle, and GreenPlum – False/Insensitive for all other services.) Both OBJECT and COLUMN are required when specifying CASE_SENSITIVITY.

Note: SCHEMA, CATALOG, DATA SERVICE, DSN, USER, and GROUP identifiers are always compared in a case insensitive manner.

Examples:

```
CREATE SERVICE ORASERV TYPE ORACLE domain ORA1 {OPTIONS ( conopts
( Driver odbc, conopts(DSN tktsora)), conopt (Driver oracle, PATH tktsora) ) }

CREATE DATA SERVICE SQLServer1 TYPE SQLSVR domain SQLSERVER {OPTIONS
( conopts ( conopts(DSN tktssql)) ) }

CREATE SERVICE DB2_SERVICE TYPE DB2UNXPC domain DB2 {OPTIONS
 ( conopts (DB DEV1) ) }
```

```
CREATE SERVICE TERA_SERVICE TYPE TERADATA domain TERA {OPTIONS
 ( conopts (Server kaching.unx.df.com) ) }

CREATE SERVICE SAPSERV TYPE SAP DOMAIN SAPDOMAIN {OPTIONS
 conopts(ashost apsrv.sup.com, sysnr 03, batch 1)}
```

DROP DATA SERVICE Statement

Enables you to drop a data service. Following is the syntax:

```
DROP [DATA] SERVICE data-service [drop-disposition]
```

data-service
Specifies the data service name.

drop-disposition
Specifies the drop disposition as one of the following values:

```
drop-disposition ::=

      {RESTRICT | CASCADE} [FORCE]
```

RESTRICT	Specifies that the drop target is empty. This is the default value.
CASCADE	Specifies that contained objects are dropped.
FORCE	Specifies the optional FORCE keyword that will suppress error messages when the data service does not exist. This additional option does not affect the performance of the RESTRICT or CASCADE options.

Examples

```
drop DATA SERVICE ORACLE3

drop service "MYSQL_SERVICE" cascade force

drop data service ORACLE1 cascade
```

ALTER DATA SERVICE Statement

Enables you to change the name of a data service. You can also change common data service attributes such as version, catalog, and domain, and other data service specific options.

```
ALTER [DATA] SERVICE data-service RENAME TO newname

ALTER [DATA] SERVICE data-service

   [ CATALOG [NAME] catalog-name ]

   [ DOMAIN [NAME] domain-name ]

   [ REGISTER [( catalog-name1 [,catalog-name2 …]) | ALL]

         [register-options]]

   [alter-data-service-options ]
```

data-service
Specifies the data service name.

newname
Specifies the new data service name.

catalog-name
Specifies the catalog name.

domain-name
Specifies the domain name.

REGISTER [(*catalog-name1* [, *catalog-name2*...]) | ALL]
Catalog registration specification. (Optional) Specify a list of catalogs to register or ALL keyword to register all catalogs visible to the connection.

```
register-options ::=
      [ UID principal ]
      [ PWD password ]
      [ VALIDATE [N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1]]
```

UID *principal*
Data Service user principal name.

PWD *password*
Data service user password.

VALIDATE [N[O] | F[ALSE] | OFF | 0 | Y[ES] | T[RUE] | ON | 1]
Specifies whether the connection and the catalog names are to be validated.

VALIDATE defaults to TRUE if not specified or if no Boolean value keyword is specified and either UID or PWD is specified. If VALIDATE is true and neither UID, nor PWD are specified, then a data service user principal name and password (personal credentials) are extracted on behalf of the caller from the domain associated with the data service. The statement returns TKTS_ERROR if no such credentials exist.

If an explicit catalog list is specified and VALIDATE is TRUE, then each catalog must exist or the statement returns TKTS_ERROR. If VALIDATE is TRUE but no catalogs are specified (including REGISTER ALL), then only the connection itself is validated since the catalog list is either empty or supplied by the connection.

alter-data-service-options
Specifies the list of data service options to alter.

```
alter-data-service-options::=

   "{" OPTIONS ["("] alter-data-service-option

               [{ "," alter-data-service-option } ...]
                                                    [")"]"}"
```

alter-data-service-option
Specifies the data service option to alter.

```
     alter-data-service-option ::=

        [ alter-operation ]data-service-option
```

data-service-option
Specifies the data service option.

```
  data-service-option::=

  conopts-configuration-list
```

conopts-configuration-list
If DRIVER driver-name is omitted, the default driver for the data service is assumed. Associated options within the CONOPTS list are used for connections using the

appropriate driver. Some data services such as ORACLE accept connections from the SAS Federation Server Driver for ODBC as well. For these data services, two CONOPTS lists can be configured, one per driver to accept connections for the two drivers. The SAS Federation Server Driver for ODBC accepts a CONOPTS driver connection string option. To configure this option and suboptions within it, the configuration option format would be CONOPTS(driver ODBC, conopts(...)). The inner CONOPTS option is a specific driver connection string (list-valued) option while the outer CONOPTS groups arbitrary driver connection string options configured for the service.

```
conopts-configuration-list::=

CONOPTS "(" [DRIVER driver-name] ["," 
driver-connection-string-option ...] ")"...
```

driver-name
Specifies the driver name.

driver-connection-string-option
Specifies the connection options that correspond to the driver which is specified in DRIVER driver-name. The DATA_SERVICE and CATALOG connection string options should not be specified here since they are implied by the data service and its configured catalogs.

Both connection options and advanced options are valid to use as driver connection string options. For more information about which connection options and advanced options are supported for each data service, see the SAS Federation Server Driver Reference on page 110 for your driver.

Autoindex ON|OFF
This option is transient and valid for the SQL_LOG data service only. Specifies whether to create indexes (ON) or to drop indexes (OFF) for the EVENTS table used for SQL Logging. The default is ON.

Examples

```
ALTER DATA SERVICE ORACLE3 {OPTIONS conopts(Driver odbc,
 ODBC_DSN tktsora)}

ALTER DATA SERVICE ORACLE3 RENAME TO ORACLE3_RENAME

ALTER DATA SERVICE ORACLE3_RENAME {OPTIONS DROP conopts(driver odbc) }

alter service service1 catalog newcatalog

alter service ORACLE4 DOMAIN ORA8

ALTER DATA SERVICE ORACLE3_RENAME {OPTIONS SET conopts(driver odbc,
 ODBC_DSN tktsora3) }

ALTER SERVICE SAPSERV {OPTIONS conopts(xset batch 1, xset destgroup SAS)};
```

CREATE CATALOG Statement

The CREATE CATALOG statement enables you to create a catalog using the syntax below.

```
CREATE CATALOG "catalog" UNDER data-service

   [ NATIVE NAME native-name ]
```

```
[ create-catalog-options ]
```

"catalog"
Specifies the catalog name.

data-service
Specifies the data service name under which the catalog is to be created.

native-name
Specifies the native catalog name. Specified when the native catalog name is not unique within the server. The native name should be used to resolve catalog name collisions between multiple data services that support catalogs. Client SQL always references the catalog via the logical catalog name (catalog) regardless of whether a native name is specified. Specifying a native name that matches the logical name does nothing.

create-catalog-options
Specifies the options to create a catalog. This option only applies to the BASE data service.

```
create-catalog-options ::=

        create-catalog-options::=

        conopts-configuration-list
```

conopts-configuration-list
If DRIVER driver-name is omitted, the default driver for the data service is assumed. Associated options within the CONOPTS list are used for connections using the appropriate driver. The multiple driver syntax is not supported

```
conopts-configuration-list::=

CONOPTS "(" [DRIVERdriver-name] ["," driver-connection-string-option ...] ")" ...
```

driver-name
Specifies the driver name.

driver-connection-string-option
Specifies the connection options that correspond to the driver which is specified in DRIVER *driver-name*.

Advanced options are valid to use as driver connection string options; connection options are invalid. For more information about which advanced options are supported for each data service, see the data service reference chapter for your driver. For information specific to the data service you are configuring, go to the section that corresponds to your data service.

Examples:

```
CREATE CATALOG "catalog1_BASE" UNDER BASE

CREATE CATALOG "TKTEST" UNDER SQLServer1

CREATE CATALOG "Catalog1" UNDER SQLServer1 NATIVE NAME "TKTEST"

CREATE CATALOG "c1" UNDER BASE {OPTIONS conopts (COMPRESS YES)}
```

DROP CATALOG Statement

The DROP CATALOG statement allows you to drop a catalog by using this syntax:

```
DROP  CATALOG "catalog" [ drop-disposition  ]
```

"catalog"
Specifies the catalog name.

drop-disposition
Specifies the drop disposition and is one of the following values:

```
drop-disposition ::=

{RESTRICT | CASCADE} [FORCE]
```

RESTRICT	Specifies that the drop target is empty. This is the default value.
CASCADE	Specifies that contained objects are dropped.
FORCE	Specifies the optional FORCE keyword that will suppress error messages when the data service does not exist. This additional option does not affect the performance of the RESTRICT or CASCADE options.

Examples:

```
drop CATALOG "Catalog3"

drop catalog "catalog1_BASE" cascade
```

ALTER CATALOG Statement

ALTER CATALOG enables you to change the name of a catalog. You can also change the native catalog name and the advanced options that are driver–specific. For information about which advanced options are supported for each data service, see the data source reference chapter for your driver.

```
ALTER CATALOG "catalog" RENAME TO "newcatalogname"

ALTER CATALOG "catalog"

   [ NATIVE NAME "native-name" ]

   [ alter-catalog-options ]
```

"catalog"
Specifies the catalog name.

"newcatalogname"
Specifies the new catalog name.

"native-name"
Specifies the name of the native catalog.

alter-catalog-options
Specifies the options to alter the catalog. This option only applies to the BASE data service. The syntax for alter-catalog-options is the same as the syntax for alter-generic-options. All create-catalog-options are also supported.

Examples:

```
ALTER CATALOG "catalog3_BASE" RENAME TO "catalog3_BASE_RENAME"

ALTER CATALOG "Catalog3" NATIVE NAME "TKTEST3_RENAME"

ALTER CATALOG "catalog1_BASE" {OPTIONS add CONOPTS(DRIVER BASE, ACCESS READONLY)}

ALTER CATALOG "catalog1_BASE" {OPTIONS set (CONOPTS(DRIVER BASE, ACCESS READONLY))}
```

```
ALTER CATALOG "catalog1_BASE" {OPTIONS xset CONOPTS(DRIVER BASE, COMPRESS YES)}

ALTER CATALOG "catalog1_BASE" {OPTIONS  drop CONOPTS(DRIVER BASE)}
```

CREATE SCHEMA Statement

Use the CREATE SCHEMA statement to create a schema and designate an owner. Here is the syntax:

```
CREATE SCHEMA [ "catalog"."schema"]

   [ AUTHORIZATION|OWNER owner ]

   [ create-schema-options ]
```

"catalog"
Specifies the optional catalog name under which to create the schema. Useful for data services defined for data sources that support catalog names. For those that do not, the catalog name must be that logical catalog name which defaults to the name of the data service.

"schema"
Specifies the schema name.

owner
Authorization identifier of the schema owner. If the AUTHORIZATION clause is not specified, then the system owns the schema. The system should never own a schema because this could cause residual problems with FedSQL views and data cache.

create-schema-options
Specifies the options to create the schema.

```
create-schema-options ::=

"{" OPTIONS ["("] schema-option

[ { "," schema-option } ... ] [")"] "}"
```

schema-option
Specifies the syntax for schema options. This option only applies to the BASE data service.

conopts-configuration-list
If DRIVER driver-name is omitted, the default driver for the data service is assumed. Associated options within the CONOPTS list are used for connections using the appropriate driver.

```
conopts-configuration-list ::=
CONOPTS "(" [DRIVER driver-name] ["," driver-connection-string-option ...] ")"...
```

driver-name
Specifies the driver name.

driver-connection-string-option
Specifies the connection options that correspond to the driver which is specified in DRIVER driver-name.

Advanced options are valid to use as driver connection string options; connection options are invalid. For more information about which advanced options are supported for each data service, see the data service reference chapter for your

driver. For information specific to the data service you are configuring, go to the section that corresponds to your data service.

PRIMARYPATH ***parh***
Specifies the physical location for the SAS library, which is a collection of one or more SAS files. For example, in directory-based operating environments, a SAS library is a group of SAS files that are stored in the same directory. This option is required for BASE schemas.

path ::=

quoted-identifier

quoted-identifier
Specifies a single quoted or double quoted name.

Examples:

```
CREATE SCHEMA "catalog1_BASE"."schema1_BASE" {OPTIONS (primarypath
'C:\schema1_BASE')}

CREATE SCHEMA "ORACLE1"."TKTSTST1"

CREATE SCHEMA "catalog1"."schema1" {OPTIONS primarypath 'C:\my_schema',
conopts (LOCKTABLE EXCLUSIVE)}
```

DROP SCHEMA Statement

Enables you to drop a schema.

```
DROP SCHEMA [ "catalog"."schema" ]  [ drop-disposition ]
```

"catalog"
Specifies the catalog name.

"schema"
Specifies the schema name.

drop-disposition
Specifies the drop disposition and is one of the following values:

drop-disposition ::=
{RESTRICT | CASCADE} [FORCE]

RESTRICT Specifies that the drop target is empty. This is the default value.

CASCADE Specifies that contained objects are dropped.

FORCE Specifies the optional FORCE keyword that will suppress error messages when the data service does not exist. This additional option does not affect the performance of the RESTRICT or CASCADE options.

Examples:

```
DROP SCHEMA "catalog1_BASE"."schema1_BASE"

DROP SCHEMA "catalog1_BASE"."schema1_BASE" force
```

ALTER SCHEMA Statement

Enables you to change the name of a schema. You can also alter advanced options that are driver–specific.

```
ALTER SCHEMA [ "catalog"."schema" ]  RENAME TO "newschema"

ALTER SCHEMA [ "catalog"."schema" ] AUTHORIZATION|OWNER TO owner

[ create-if option ]

ALTER SCHEMA [ "catalog"."schema" ]

[ alter-schema-options ]
```

"catalog"
Specifies the catalog name.

"schema"
Specifies the schema name.

"newschema"
Specifies the new schema name.

alter-schema-options
Specifies the options to alter the schema.

```
 alter-schema-options ::=

   "{" OPTIONS ["("] alter-schema-option

[{"," alter-schema-option} ... ] [")"]"}"
```

alter-schema-option
Specifies the schema option to alter. This option only applies to the BASE data service.

```
 alter-schema-option ::=

[DROP schema-option-name ]
     [{ADD | SET} schema-option ]
       [create-if-option
```

schema-option
Specifies the syntax for schema options.

conopts-configuration-list
If DRIVER *driver-name* is omitted, the default driver for the data service is assumed. Associated options within the CONOPTS list are used for connections using the appropriate driver.

```
conopts-configuration-list ::= CONOPTS "(" [DRIVER driver-name]
 ["," driver-connection-string-option ...] ")" ...
```

driver-name
Specifies the driver name.

driver-connection-string-option
Specifies the connection options that correspond to the driver which is specified in DRIVER *driver-name*.

Advanced options are valid to use as driver connection string options; connection options are invalid. For more information about which advanced options are

supported for each data service, see the data service reference chapter for your driver. For information specific to the data service you are configuring, go to the section that corresponds to your data service.

PRIMARYPATH ***path***
Specifies the physical location for the SAS library, which is a collection of one or more SAS files. For example, in directory-based operating environments, a SAS library is a group of SAS files that are stored in the same directory. This option applies to BASE schemas only and is required.

```
path ::=    quoted-identifier
```

quoted-identifier Specifies a single quoted or double quoted name.

create-if-option
Creates the schema if it does not already exist using the remaining options.

```
create-if-option
CREATE_IF N[O]|F[ALSE]|OFF|0|Y[ES]|T[RUE]|ON|1
```

Examples:

```
ALTER SCHEMA
"catalog1_BASE"."schema3_BASE" RENAME TO "schema3_BASE_RENAME"

ALTER SCHEMA "catalog1_BASE"."schema3_BASE" {OPTIONS set primarypath 'C:\mydir'}

ALTER SCHEMA "catalog1_BASE"."schema3_BASE" {OPTIONS add conopts (LOCKTABLE SHARE)}
```

CREATE DSN Statement

The CREATE DSN statement enables you to create a standard, single-service DSN or a federated DSN. A federated DSN is created to group one or more standard DSNs.

For additional information on standard and federated DSNs, see the Configuring DSNs on page 86 topic.

Standard DSN

```
CREATE DSN dsn-name

UNDER data-service

[DESCRIPTION 'description text']

[CONNECT 'driver-connection-string-options']

[create-dsn-options]

[AS ADMINISTRATOR]
```

Federated DSN

```
CREATE DSN dsn-name

[DESCRIPTION 'description text']

[create-dsn-options]

ADD "(" dsn-name
["," ...] ")"

      [AS ADMINISTRATOR]
```

dsn-name

Specifies the DSN name (required). Quotation marks surrounding the DSN name are optional.

```
CREATE DSN dsn-name
```

data-service

Specifies the data service name (required).

```
UNDER data-service-name
```

The following rules apply:

- The specified name must not match the reserved names of the internal BASE data service.
- The specified name must not match the name of any existing data service.
- The specified name must not match the name of an existing catalog for defined data services that do not support catalog names, unless the CATALOG option is used to specify a different name.

DESC[RIPTION] *'description-text'*

Description of the DSN surrounded in single quotation marks. Use for a standard or federated DSN (optional).

```
[DESC 'description text']
```

CONNECT *driver-connection-string-options*

Specifies the connection string options.

```
[CONNECT driver-connection-string-options']
```

The Federation Server driver connection string options are an extension of the ODBC syntax that specifies options as semicolon-delimited key=value pairs. For more information about which connection options and advanced options are supported for each data service, see the driver reference topic on page 110 for your data source driver.

AS ADMINISTRATOR

Creates the DSN using the ADMINISTRATOR role as the owner. With the ADMINISTRATOR role, the DSN is owned by the individual user. If the user is SYSTEM, the DSN is owned by SYSTEM. 'AS ADMINISTRATOR ' is optional and can be used in a standard or federated DSN.

```
[AS ADMINISTRATOR]
```

DSN Options

create-dsn-options

Specifies the common DSN option as one of the following:

dsn-config-options

Specifies the options to configure the DSN.

```
dsn-config-options ::=
 "{" OPTIONS ["("] dsn-config-option
      [{"," dsn-config-option} ... ] [")"]"}"
```

dsn-config-option

Specifies the DSN configuration option as one of the following:

FEDSQL Y[ES] | T[RUE] | 1 | N[O] | F[ALSE] | 0

Specifies whether to use the FedSQL dialect. Base DSN connections always use FedSQL. Therefore, FedSQL cannot be turned off for BASE DSNs. The FEDSQL option applies to both standard and federated DSNs.

SECURITY Y[ES] | T[RUE] | 1 | N[O] | F[ALSE] | 0

YES is the default value. Specifies whether to secure SQL statements before processing them. For example, if a DSN is defined to use SECURITY NO, Federation Server security is bypassed. Therefore, when you connect with the DSN, you are connecting with the privileges granted at the data source level. If a DSN is defined to use SECURITY YES, privileges granted through the Federation Server will be enforced in addition to those of the underlying data source. Used in conjunction with CSO SHARED, this feature facilitates management of a more granular security policy in the Federation Server over a less granular one in the back-end database.

If SECURITY is set to YES, FEDSQL is automatically set to YES.

The SECURITY option applies to both standard and federated DSNs. It corresponds to the Federation Server SQL Authorization Enforcement setting that is displayed for the DSN through the Federation Server Manager.

On a standard DSN, SECURITY NO will ignore any SQL privileges configured in the Federation Server. SECURITY YES enforces SQL privileges. See Security Permissions in the Federation Server Authorization section for a list of privileges that are affected by the SECURITY setting.

On a federated DSN, SECURITY NO indicates that the security setting of the child DSN is used. A setting of NO allows each child DSN to operate under the security settings that are configured for it. Setting SECURITY to YES activates SQL privilege enforcement for all of the child DSNs affiliated with the federated DSN. Effectively, the SECURITY setting on a federated DSN can demand privilege enforcement on child DSNs, but cannot be used to remove it.

CREDENTIALS_SEARCH_ORDER | CSO "(" *cso-value* [{"," *cso-value*} ...] ")"

Specifies whether to use back-end credentials owned by the current user (PERSONAL) or shared among many users (SHARED). The DSN can be configured to search for either in the order specified. If a search is not specified, the default is CSO (PERSONAL,SHARED). CSO applies to a standard DSN only.

For example, if a user owns a database login and a DSN is configured with a CSO value of PERSONAL, the DSN will use that user's database login to connect to the database. However, when the DSN is configured using CSO SHARED and the user is configured as an authorized consumer of a shared login, the DSN connects using the shared login credentials. Users, logins, and shared logins are created and managed using the Authentication Server. For more information, see the *DataFlux Authentication Server User's Guide*.

```
cso-value ::= PERSONAL |SHARED
```

Examples:

```
CREATE DSN "DSN1" UNDER BASE DESCRIPTION 'creating DSN1' NOPROMPT
'DRIVER=BASE;CATALOG="catalog1_BASE";SCHEMA="schema1"' {OPTIONS (FEDSQL
NO,SECURITY NO)}

CREATE DSN BASEDSN under BASE NOPROMPT 'DATA_SERVICE=BASE;
LOCKTABLE=EXCLUSIVE'

CREATE DSN BASEDSN under BASE CONNECT 'CATALOG="catalog1_BASE";
LOCKTABLE=EXCLUSIVE'

CREATE DSN BASEDSN under BASE NOPROMPT '(CATALOG="catalog1_BASE";
LOCKTABLE=SHARE);(CATALOG="catalog2_BASE";LOCKTABLE=EXCLUSIVE)'
```

```
CREATE DSN BASEDSN under BASE NOPROMPT 'CATALOG="catalog1_BASE";
LOCKTABLE=SHARE;SCHEMA=(NAME="schema1_BASE";LOCKTABLE=EXCLUSIVE)'

CREATE DSN BASEDSN under BASE CONNECT 'CATALOG="catalog1_BASE";
LOCKTABLE=SHARE;SCHEMA=(NAME="schema1_BASE";LOCKTABLE=EXCLUSIVE);
SCHEMA=(NAME="schema2_BASE";ACCESS=TEMP)'

CREATE DSN MYDSN under MYSERV {OPTIONS CREDENTIALS_SEARCH_ORDER
(PERSONAL)}

CREATE DSN MYDSN under MYSERV {OPTIONS CREDENTIALS_SEARCH_ORDER
(PERSONAL, SHARED)}

CREATE DSN ORADSN UNDER ORASERVICE CONNECT 'ORA ENCODING=UNICODE;
ORANUMERIC=YES'

CREATE DSN MYFEDERATED_DSN ADD mydsn1, mydsn2, mydsn3)
AS ADMINISTRATOR
```

DROP DSN Statement

The DROP DSN statement enables you to drop a server-based DSN.

```
DROP DSN dsn-name [FORCE]
```

dsn-name
Specifies the DSN name.

FORCE Specifies the optional FORCE keyword that will suppress error messages when the DSN does not exist.

.

Examples:

```
DROP DSN "DSN1"

DROP DSN "DSN1" FORCE
```

ALTER DSN Statement

Enables you to change the server–based DSN. You can change the name and alter advanced options. For information about which advanced options are supported for each data source, see the Driver Reference chapter for your driver.

Syntax

```
ALTER DSN dsn-namealter-dsn-options

ALTER DSN dsn-name RENAME TO new-dsn-name

ALTER DSN dsn-name ADD "(" dsn-name ["," ...] ")"

ALTER DSN dsn-name DROP "(" dsn-name ["," ...] ")"
```

dsn-name
Specifies the DSN name.

alter-dsn-options
Specifies the options to alter.

```
alter-dsn-options

   ::= create-dsdsn-options
```

new-dsn-name
Specifies the new DSN name.

Examples:

```
ALTER DSN "DSN1" DESC 'altering DSN1 description' NOPROMPT
'DRIVER=BASE;CATALOG="catalog1_BASE";SCHEMA=(name="schema1_BASE")'

ALTER DSN "DSN5" RENAME to DSN7

ALTER DSN "DSN7" {OPTIONS set (FEDSQL
YES,SECURITY YES)}

ALTER DSN "DSN7" {OPTIONS xset CREDENTIALS_SEARCH_ORDER(SHARED),
 xset FEDSQL NO, xset SECURITY NO}

ALTER DSN "DSN7" {OPTIONS DROP FEDSQL, DROP SECURITY}
```

CREATE CACHE Statement

This statement creates a cache definition for the specified view in the cache catalog and schema with the specified options.

```
CREATE CACHE "catalog"."schema"."view"

IN "cache-catalog"."cache-schema"

[{OPTIONS SAS-table-option=value [ ... SAS-table-option=value ]}]

[USING ( fedsql_syntax_sql )]

[WITH COMMENT '…']

[DEFERRED]

[FORCE]

[EXEC|EXECUTE]

[BEFORE  (fedsql_syntax_sql) [FORCE]

       [,(fedsql_syntax_sql) [FORCE] […]]]

[AFTER  (fedsql_syntax_sql) [FORCE]

       [,(fedsql_syntax_sql) [FORCE] […]]]

[CLEANUP  (fedsql_syntax_sql) [FORCE]

       [,(fedsql_syntax_sql) [FORCE] […]]]

[SET] <option_name> = '<string_literal>' | <numeric_literal>

           | ON | OFF | YES | NO | TRUE | FALSE ]
```

"catalog"
Specifies the catalog name of the view to be cached.

"schema"
Specifies the schema name of the view to be cached.

"view"
Specifies the name of the cache view.

"cache-catalog"
Specifies the catalog name under which to create the cache.

"cache-schema"
Specifies the schema name under which to create the cache.

{OPTIONS SAS-table-option=*value*... }
Specifies the table options to use when the data cache table is created and populated.

[USING (fedsql_syntax_sql)]
The optional USING clause provides a way for users to control how the cache table is created. The user must ensure that it is compatible with the view data cache table.

WITH COMMENT '...'
Text comments stored with the cache definition.

DEFERRED
The cache definition is stored but a cache table is not created or populated with data. A separate REFRESH CACHE command is needed to create and populate the cache table.

FORCE
The cache table and definition is retained even if an error occurs during REFRESH.

EXEC | EXECUTE
Command used with BEFORE, AFTER, and CLEANUP options.

BEFORE	The statements in fedsql_syntax will be executed before the cache table is created and populated. The FORCE option will suppress any errors.
AFTER	The statements in fedsql_syntax will be executed after the cache table is created and populated. The FORCE option will suppress any errors.
CLEANUP	The statements in fedsql_syntax will be executed in the event of an error during creation or population of the cache table. The FORCE option will suppress any errors.

[SET] <*option name*> = <*value*>
Specifies additional options and values to use during cache creation and population. The options are:

ERRLMIT	Sets a limit on the number of errors to allow before a statement stops inserting data.
DBCOMMIT	Sets a limit on the number of modified rows to commit at one time, which affects transaction logging limits on the back-end database. This option overrides the ERRLIMIT option.
INSERTBUFF	Sets a limit for the number of rows inserted at a time which places a limit on a driver's row array size when inserting data.
CT_PRESERVE	Sets the CT_PRESERVE connection string option which controls how data types are mapped between the source view and the cache table.

Notes:

Use the following keywords with the USING, BEFORE, AFTER, and CLEANUP clauses only. The keywords must be UPPER CASE with no blank spaces within the braces {....}.

{CACHE}
Expands to fully qualified cache table name using quotation marks, for example: “catalog”.”schema”.”cache-table”

{CACHE_CATALOG}
Expands to cache catalog name. Do not use quotation marks.

{CACHE_SCHEMA}
Expands to cache schema name. Do not use quotation marks.

{CACHE_TABLE}
Expands to cache table name. Do not use quotation marks.

Examples:

```
CREATE CACHE ORACLE_SERVICE.TKTSTST1.view_red in "SAMPLE"."tktstst1" set
CT_PRESERVE='SAFE'
```

REFRESH CACHE Statement

The REFRESH CACHE statement refreshes the data cache on a specified view.

Syntax

```
ALTER CACHE "catalog"."schema"."view" REFRESH
```

“catalog”.”schema”.”view”
Specifies the catalog and schema of the data cache view.

REFRESH Refreshes cache of the specified cache view.

ALTER CACHE Statement

With the ALTER CACHE statement, you can disable, enable or refresh cache tables. This requires the CREATE CACHE or ALTER CACHE privilege.

```
ALTER CACHE "catalog"."schema"."view"OPTION
```

“catalog”.’schema”.”view”
Specifies the catalog, schema and name of the view.

OPTION
Option for the statement. One of the following:

DISABLE Disables use of the specified cache. References to the view will use the view rather than the cached data.

ENABLE Enables use of the specified cache.

REFRESH Refreshes the cache table for the specified view.

Examples

```
ALTER CACHE "catalog1"."schema1"."view1" DISABLE
```

```
ALTER CACHE [ "catalog2"."schema2". ] view2 ENABLE

ALTER CACHE [ "catalog3"."schema3". ] view3 REFRESH
```

DROP CACHE Statement

Enables you to drop a cached view.

```
DROP CACHE [ "catalog"."schema". ] view [FORCE ]
```

"catalog"."schema"
Specifies the catalog and schema of the data cache view.

FORCE Suppresses error messages if the cache to be dropped does not exist.

.

PURGE CACHE Statement

PURGE CACHE forces the removal of outdated cache tables that are no longer needed. PURGE CACHE is also used with options in the ALTER SERVER statement. Only system users or those with ADMINISTER privilege on the SAS® Federation Server can execute PURGE CACHE through the ALTER SERVER statement.

```
PURGE CACHE
```

DROP AUTHID Statement

Enables you to drop authorization identifiers.

```
DROP { AUTHID | AUTHORIZATION [IDENTIFIER] } "ID" [TRANSFER TO
name] [CASCADE|RESTRICT] [FORCE]
```

"ID"
Specifies the authorization identity to drop. The ID must be in quotation marks.

This is the user ID or the group ID that you can find by using the AUTHORIZATION_IDENTIFIERS information view.

name
Specifies the authorization identity that will receive object ownership from the dropped identity. This user name is created using the Authentication Server.

FORCE Specifies the optional FORCE keyword that will suppress error messages when the user does not exist

CASCADE Specifies the entities are dropped unconditionally. All records that reference the entity are removed. This option is invalid if the TRANSFER TO option is specified.

RESTRICT Specifies the drop fails if the entity is the grantor of any privilege. The drop also fails if the entity is the owner of a DSN or schema. This option is ignored if the TRANSFER TO option is specified.

Examples:

```
drop authid "5E563F78B0D70854086FB3D8441EF9AA" transfer to user2

drop authid "F135005B80DED494E996F70DCC53790D"  cascade

drop authid "B8A105927F25B1A47AE8198D1E3C4B86" transfer to user2 force
```

GENERIC OPTIONS Syntax

Generic options is a set of comma separated name or name-value pairs, within an OPTIONS list.

```
generic-options ::=

      "{" OPTIONS ["("] generic-option-list [")"] "}"

generic-option-list ::=

      generic-option[ {"," generic-option} ... ]

generic-option ::=

      option-name [ {option-value | option-list} ]


option-list ::=

      "("
option-value [ {"," option-value} ... ] ")"

option-value ::=
     quoted-identifier
```

ALTER GENERIC OPTIONS Syntax

A set of comma separated name or name-value pairs with optional operation keywords, within an OPTIONS list.

```
alter-generic-options ::=
"{" OPTIONS ["("] alter-generic-option-list [")"] "}"

alter-generic-option-list ::=
alter-generic-option [ {"," alter-generic-option} ... ]

alter-generic-option ::=
[alter-operation ]option-name
[ {option-value |option-list } ]

alter-operation ::=
ADD | SET | XSET | DROP
```

The possible values for the alter-operation option are:

ADD	Adds the option. Add is the default if an option is not specified.
SET	Changes the option that already exists.
XSET	Sets the option, if it has already been added. Or, adds the option if it does not already exist.

DROP Drops the option.

Note: If a value is omitted for alter-operation, the default operation is ADD.

GRANT and DENY Statements

Enables you to give privileges to a specific user or all users to perform actions on objects. When submitting a grant, revoke, or deny DDL, surround all identifiers in quotation marks, including table and column names.

Note: CREATE DSN and ADMINISTER privileges cannot be granted or denied for the PUBLIC and USERS groups.

Syntax

```
GRANT | DENY { {"objectpriv" |"containerpriv" |
"serverpriv" [,...] } |

               ALL [ PRIVILEGES ] }

   [ ON { SCHEMA "schemaname" | CATALOG "catalogname" |
 [DATA] SERVICE "servicename" | DSN "dsnname" | SERVER } ]

   TO { "authid" | PUBLIC | USERS } [, ...]

    [ AS ADMINISTRATOR ]
```

"objectpriv"
Specifies the name of an object-level privilege to grant or deny, as one of the following values:

- SELECT
- INSERT
- UPDATE
- DELETE
- REFERENCES

"containerpriv"
Specifies the name of a container-level privilege to grant or deny, as one of the following values:

- CREATE VIEW
- ALTER VIEW
- DROP VIEW
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

"serverpriv"
Specifies the name of the server-level privilege to grant or deny, as one of the following values:

- ADMINISTER
- TRACE

- CREATE DSN
- CONNECT

"schemaname"
Specifies the name of the schema.

"catalogname"
Specifies the name of the catalog.

"servicename"
Specifies the name of the data service.

"dsnname"
Specifies the name of the DSN.

"authid"
Specifies the user or group name for which the privileges are granted or denied.

AS ADMINISTRATOR
Grants privileges using the ADMINISTRATOR role. Without this, the privilege is granted as the individual user.

Note: If the user is a system user, then privileges are assigned with SYSTEM as the grantor.

Examples:

```
GRANT INSERT ON SCHEMA "BASE_CATALOG1"."schema1_BASE" TO "user1"

GRANT CONNECT ON SERVER TO "user1"

DENY CONNECT ON DSN "SQLSRVDSN1" TO "user1"

GRANT CREATE DSN ON DATA SERVICE "SQLSRV1" TO "user1"

GRANT ADMINISTER ON SERVER TO "user1"

DENY ALL ON SCHEMA "BASE_CATALOG1"."schema1_BASE" TO "user1"
```

REVOKE Statement

Enables you to remove explicitly granted or denied privileges from the specified object.

```
REVOKE { { "objectpriv" | "containerpriv" |
"serverpriv" [,...] } |

ALL [ PRIVILEGES ] }

[ ON { SCHEMA "schemaname" | CATALOG "catalogname" |
     [DATA] SERVICE "servicename" | DSN "dsnname" | SERVER} ]

FROM { "authid" | PUBLIC | USERS } [, ...]
```

"objectpriv
Specifies the name of an object–level privilege to grant or deny, as one of the following values:

- SELECT
- INSERT
- UPDATE
- DELETE

- REFERENCES

"containerpriv"
Specifies the name of a container–level privilege to grant or deny, as one of the following values:

- CREATE VIEW
- ALTER VIEW
- DROP VIEW
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

"serverpriv"
Specifies the name of the server–level privilege to grant or deny, as one of the following values:

- ADMINISTER
- TRACE
- CREATE DSN
- CONNECT

"schemaname"
Specifies the name of the schema.

"catalogname"
Specifies the name of the catalog.

"servicename"
Specifies the name of the data service

"dsnname"
Specifies the name of the DSN.

"authid"
Specifies the user or group name for which the privileges are granted or denied.

AS ADMINISTRATOR
Grants privileges using the ADMINISTRATOR role. Without this, the privilege is granted as the individual user.

Note: If the user is a system user, then privileges are assigned with SYSTEM as the grantor.

Examples:

```
REVOKE ALL ON SCHEMA "BASE_CATALOG1"."schema1_BASE" FROM "user1"

REVOKE INSERT ON DATA SERVICE BASE FROM "USER1"

REVOKE all on server from "user1"
```

Appendix 2

Information Views

Visibility Rules for Information Views

Data visible in the information views is based on the user object and the privileges associated with the object. Therefore, user privileges determine what records are visible to the user. All users can query views but an empty result set is returned if the user does not have privileges to a specific view. Use the ADMIN DSN to connect to the information views.

A majority of the information views return system-level data that is relevant only to administrators or to technical support staff working with customers. The exceptions are certain information views that return privilege information, since users should be able to see what privileges they are granted on objects for which they have at least a single privilege.

The following information summarizes user visibility and the data returned from SAS Federation Server.

Administrators and System Users

System users and server administrators can view all data in all information views. The following related views are restricted to system users and administrators only:

Table A2.1 *Administrators and System Users*

Views	Visibility
AUTHORIZATION_IDENTIFIERS OBJECTS COLUMNS	System user and SAS® Federation Server administrators only

Data Services

The following table lists the visibility rules that are associated with information views that are related to data services:

Table A2.2 *Data Services*

Views	Visibility
DATA_SERVICES CONFIG_DATA_SERVICES	A data service is visible to a user if: • the user has CONNECT, ADMINISTER, or CREATE DSN privileges on the data service, or • the user has CONNECT privilege on any data service DSN.

DSN

The following table lists the visibility rules that are associated with information views for data sources names:

Table A2.3 *DSN*

Views	Visibility
DATA SOURCE NAMES DSN_CONTENT DSN_LINEAGE CONFIG_DSNS	A data source name (DSN) is visible to a user if: • the user is the owner of the DSN, or • has CONNECT privilege on the DSN.

Catalogs and Schemas

SAS Federation Server needs to display catalogs and schemas for the BASE service without connecting to the data service first. This is different from other data services because SAS Federation Server Manager can connect to a data service and query it for an associated list of catalogs and schemas. Non-administrator users must be able to see BASE objects. One example is if the user has CREATE CACHE privilege and needs to be able to cache views from the user interface. Creating views from SAS Federation Server Manager is another example. Results from the catalogs and schemas information views will be filtered depending on the user's privileges.

Table A2.4 *Catalogs and Schemas*

Views	Visibility
CATALOGS CONFIG_CATALOGS SCHEMAS CONFIG_SCHEMAS	A catalog is visible to a user if: • the data service is visible. A schema is visible to a user if: • the data service is visible.

Object Privileges

The following table lists the visibility rules that are associated with information views for object privileges:

Table A2.5 *Object Privileges*

Views	Visibility
DSN_PRIVILEGES DS_PRIVILEGES CATALOG_PRIVILEGES SCHEMA_PRIVILEGES OBJECT_PRIVILEGES COLUMN_PRIVILEGES	Privilege rows are visible to a user if: • the user is the grantor of the privilege, or • the user is the grantee of the privilege, or • one of the user's groups is the grantee of the privilege (including the USERS or PUBLIC group) AND • the user has at least one privilege on the object in the view (DSN/data service/catalog/schema/object/column)

Data Cache

Data cache metadata is distributed between the CACHES, MESSAGES and CONFIG_OBJECTS information views. Users with CREATE CACHE or ALTER CACHE privilege will need to see data from these information views.

Table A2.6 *Data Cache*

Views	Visibility
CACHES MESSAGES CONFIG_OBJECTS	Data items are visible to a user if: • the item is a data cache item, and • the user has CREATE CACHE or ALTER CACHE privilege on the item

Container and Object Privileges

Privileges in the container and object categories pertain to server, data services, catalogs, schemas, objects, and columns.

Table A2.7 *Container and Object Privileges*

Views	Visibility
DSN_PRIVILEGES and EFFECTIVE_DSN_PRIVILEGES PRIVILEGES and EFFECTIVE_PRIVILEGES X_COLUMN_PRIVILEGES/ X_EFFECTIVE_COLUMN_PRIVILEGES	Privileges for these items are visible to a user if: • the user is the grantee of the privilege. • one of the user's groups is the grantee of the privilege, including the USERS group. • the privilege is granted in the PUBLIC group.

AUTHORIZATION_IDENTIFIERS View

The AUTHORIZATION_IDENTIFIERS view displays Authentication Server objects (users and groups) that have been resolved on SAS Federation Server. This view also shows inactive records for SAS Federation Server which are records created when an object is removed from the Authentication Server. Removing an object from the Authentication Server does not remove it from the AUTHORIZATION_IDENTIFIERS view. Use the DROP AUTHID | AUTHORIZATION IDENTIFIER on page 181 DDL statement to remove these objects from the AUTHORIZATION_IDENTIFIERS view.

The following table lists the columns that will be displayed:

Name	Type	Description
NAME	VARCHAR(256)	Specifies the name of a group, role, or user that was created using the Authentication Server. NULL is displayed if the group, role, or user is orphaned.
ID	VARCHAR(256)	Specifies the authorization identifier from the Authentication Server.

Name	Type	Description
TYPE	CHAR(1)	Specifies the type of entity as G (Group) or U (User).

Only System or Federation Server administrators can view data in the AUTHORIZATION_IDENTIFIERS view. All users can query the AUTHORIZATION_IDENTIFIERS view, but the query will return an empty result set if the user is not a system user or SAS Federation Server administrator.

CACHES View

The CACHES view displays information for all defined caches. The table below lists the columns that are associated with the CACHES view:

Name	Type	Description
CATALOG_NAME	WVARCHAR(256) Not Null	Specifies the catalog containing the object being cached.
SCHEMA_NAME	WVARCHAR(256) Not Null	Specifies the schema containing the object being cached.
OBJECT_NAME	WVARCHAR(256) Not Null	Specifies the name of the object being cached.
CACHE_STATUS	WCHAR(1) Not Null	Specifies the current status of the cache: E – An ERROR row indicates that the last operation failed - there should only ever be one ERROR row (or 0 if the last command succeeded). D - A DEFERRED row indicates that the last operation was specified as DEFERRED and has not yet been successfully refreshed - there should only ever be one DEFERRED row (or 0 if the view was successfully refreshed). A – An ACTIVE row indicates that valid or complete cache data for the object exists, and is available for use. New users should always "see" the ACTIVE row with the latest timestamp. Old ACTIVE rows will go away as their user's pending transactions end. At start-up, only the newest ACTIVE row will remain. S - SUSPENDED indicates that a cache has been disabled.
START_TS	TIMESTAMP Nullable	Specifies when the refresh cache operation began.
END_TS	TIMESTAMP Nullable	Specifies when the refresh cache operation completed.
MESSAGE_ID	INTEGER Nullable	Shows any message generated when the cache was defined or refreshed. (See the “MESSAGES View” on page 203.
NUM_ROWS	BIGINT Nullable	Specifies the number of rows inserted into the data cache table. Note that this should be considered a rough estimate.

Name	Type	Description
NUM_BYTES	BIGINT Nullable	Specifies the number of bytes inserted into the data cache table. Note that this should be considered a rough estimate.
USER_NAME	WVARCHAR(256)	Shows the user who performed the CREATE CACHE or REFRESH cache operation that caused this row.

Note: Only FedSQL views with definer's rights can be cached.

Status Rules for rows in the CACHES Table

The most recent row for a cache is returned, no matter the status. If the most recent row was Active, that should be the only row returned. If the most recent row was Deferred or Error, the next most recent Active row is returned if one exists.

CATALOGS View

The CATALOGS view contains the name and system identifier for each catalog.

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog.
DATA_SERVICE_NAME	VARCHAR(256)	Specifies the name of the data service associated with catalog.
NATIVE_CATALOG_NAME	VARCHAR(256)	Specifies the name of the native catalog. If not applicable, the value is NULL.

CATALOG_PRIVILEGES View

The CATALOG_PRIVILEGES view displays the catalog-level privileges for each catalog. The table below lists the columns that are associated with the CATALOG_PRIVILEGES view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	The catalog name.
GRANTOR	VARCHAR(256)	The user name of the grantor.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as R (Role) or U (User).
GRANTEE	VARCHAR(256)	Specifies the user name of the grantee.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as G (Group) or U (User).

Name	Type	Description
PRIVILEGE_NAME	VARCHAR(20)	Privilege name specified as one of the following: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW CREATE CACHE ALTER CACHE CREATE TABLESPACE
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege can be granted to others. The only valid value is N (No).

COLUMNS View

The COLUMNS view contains the name and system identifier for each column. The table below lists the columns that are associated with the COLUMNS view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the catalog name.
SCHEMA_NAME	VARCHAR(256)	Specifies the schema name.
OBJECT_NAME	VARCHAR(256)	Specifies the object name.
COLUMN_NAME	VARCHAR(256)	Specifies the column name.

COLUMN_PRIVILEGES View

The COLUMN_PRIVILEGES view displays the privilege descriptors for all column-level privileges. The table below lists the columns that are associated with the COLUMN_PRIVILEGES view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog.
SCHEMA_NAME	VARCHAR(256)	Specifies the name of the schema.
OBJECT_NAME	VARCHAR(256)	Specifies the name of the object.
COLUMN_NAME	VARCHAR(256)	Specifies the column name.
GRANTOR	VARCHAR(256)	Specifies the user name of the grantor.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE	VARCHAR(256)	Specifies the name of the user who is granted privileges.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as G (Group) or U (User).
PRIVILEGE_NAME	VARCHAR(20)	Specifies the privilege name as one of the following values: SELECT UPDATE INSERT REFERENCES
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege is grantable. The only valid value is N (No).

CONFIG_CATALOGS View

The CONFIG_CATALOGS view contains generic configuration variables for each defined catalog. All configuration settings for a single catalog can be obtained by concatenating rows matching the correct CATALOG_NAME and ordering the results by sequence.

The table below lists the columns that are associated with the CONFIG_CATALOGS view:

Name	Type	Description
DATA	VARCHAR(128)	Specifies the configuration data as not NULL.
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog.
SEQUENCE	SMALLINT	Specifies the configuration chunk sequence.

CONFIG_DATA_SERVICES View

The CONFIG_DATA_SERVICES view contains generic configuration variables for each defined data service. All configuration settings for a single service can be obtained by concatenating rows matching the correct DATA_SERVICE_NAME and ordering the results by sequence.

The table below lists the columns that are associated with the CONFIG_DATA_SERVICES view:

Name	Type	Description
DATA	VARCHAR(128) Not Null	Specifies the configuration data.
DATA_SERVICE_NAME	VARCHAR(256)	Specifies the unique name of the data service.
SEQUENCE	SMALLINT	Specifies the configuration chunk sequence.

CONFIG_DSNS View

The CONFIG_DSN view displays generic configuration variables for each DSN defined in the definition schema. All configuration settings for a single DSN can be obtained by concatenating rows that match the correct DSN_NAME and ordering the results by sequence.

The table below lists the columns that are associated with the CONFIG_DSNS view:

Name	Type	Description
DATA	VARCHAR(128) Not Null	Specifies the configuration data.
DSN_NAME	VARCHAR(256)	Specifies the unique name of the DSN.
SEQUENCE	SMALLINT	Specifies the configuration chunk sequence.

CONFIG_OBJECTS View

The CONFIG_OBJECTS view contains the configuration statement for the specified object. Stored information varies by object. The configuration statement text is broken up into pieces by type based on how FedSQL parses the statement. Sub-text for each type is broken up to fit into the DATA column ordered by CFG_TYPE_SEQUENCE.

Concatenating the DATA entries for a given OBJECT_NAME in SEQUENCE order will produce the original configuration statement syntax with the following exceptions:

- It will be normalized and broken up into pieces based on what options are specified.
- It can be reordered, although positional clauses like “EXEC” will always remain in sequence
- It will contain any comments that were in the original create cache statement that was submitted
- It might be modified to contain information specified by later statements (like ALTER).

Note: A view cache will return a single entry containing the most recent CREATE CACHE statement, an OBJECT_TYPE of 2 (SAS View) or 3 (CACHE), and the catalog/schema/name of the view being cached.

The following table lists the available columns for this view. Data is visible only if the user has CREATE CACHE or ALTER CACHE privilege on the referenced view.

Name	Type	Description
CATALOG_NAME	WVARCHAR(256) Not Null	Specifies the catalog containing the object.
SCHEMA_NAME	WVARCHAR(256) Not Null	Specifies the schema containing the object.
OBJECT_NAME	WVARCHAR(256) Not Null	Specifies the object name.
OBJECT_TYPE	INTEGER Not Null	Specifies the object type: 1 – Table, 2 – SAS View, 3 – Cache, 4 – SAS Package
DATA	WVARCHAR(128) Not Null	A piece of text from the configuration statement.
SEQUENCE	INTEGER Not Null	Orders the entries for the entire configuration statement for a single object.
CFG_TYPE	INTEGER Not Null	Indicates the type for this piece of the configuration statement as parsed by FedSQL.
CFG_TYPE_SEQUENCE	INTEGER	Orders the entries within each type.

CONFIG_SCHEMAS View

The CONFIG_SCHEMAS view contains generic configuration variables for schemas defined in the definition schema. All configuration settings for a single schema can be obtained by concatenating rows matching the correct CATALOG_NAME and SCHEMA_NAME and ordering the results by sequence.

The table below lists the columns that are associated with the CONFIG_SCHEMAS view:

Name	Type	Description
DATA	VARCHAR(128) Not Null	Specifies the configuration data.
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog. If not applicable, the value is NULL.
SCHEMA_NAME	VARCHAR(256)	Specifies the name of the schema.
SEQUENCE	SMALLINT	Specifies the configuration chunk sequence.

DATA_SERVICES View

The Data Services view displays information about each data service. It also shows a single entry for SAS Federation Server with a value in the data_service_name column of _SERVER_ and a value in the type column of SERVER.

The DATA_SERVICES table contains one entry per configured data service, both internal and external. The table below lists the columns that are associated with the DATA_SERVICES view:

Name	Type	Description
DATA_SERVICE_NAME	VARCHAR(256)	The unique name of the data service.
VERSION	CHAR(32)	The version of the data service.

Name	Type	Description
TYPE	CHAR(32)	Keyword for the data type. Valid values include: BASE DB2 GREENPLUM MDS ODBC ODBC_FED ORACLE TRAN TERADATA SQLSVR Other values are possible.
DOMAIN	VARCHAR(256)	The Authentication Server domain that is associated with the data service.

DATA_SOURCE_NAMES View

The DATA_SOURCE_NAMES view contains one entry per configured DSN and includes the following:

- This view contains a default BASE DSN.
- This view also contains an entry for the ADMIN DSN which is a DSN generated by the system to be used with server administration DDL and system catalog queries. The value of the DATA_SERVICE_NAME column is _SERVER_.
- If SQL Logging is enabled, this view also shows the SQL_LOG DSN.

The table below lists the columns that are associated with the DATA_SOURCE_NAMES view:

Name	Type	Description
DSN_NAME	VARCHAR(256)	Specifies the unique name of the DSN.
DATA_SERVICE_NAME	VARCHAR(256)	Specifies the unique name of the data service.
DESC	VARCHAR(256)	Specifies the descriptive text.
FORMAT	CHAR(32)	Specifies the format of the content as STANDARD, which is the standard driver connection string.
OWNER_NAME	VARCHAR(256)	Specifies the name of the user that owns the DSN.

Name	Type	Description
OWNER_ID	VARCHAR(256)	Specifies the ID of the user that owns the DSN.
OWNER_TYPE	CHAR(1)	Specifies the owner type as U (User) or R (Role).
DSN_TYPE	VARCHAR(256)	Specifies the DSN type as one of the following: FEDERATED NOPROMPT CONNECT FILE

DS_PRIVILEGES View

The DS_PRIVILEGES view displays the privilege descriptors for all data source-level privileges.

The table below lists the columns that are associated with the DS_PRIVILEGES view:

Name	Type	Description
DATA_SERVICE_NAME	VARCHAR(256)	Specifies the unique name of the data service.
GRANTOR	VARCHAR(256)	Specifies the user name of the grantor.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE	VARCHAR(256)	Specifies the user name of the grantee.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).

Name	Type	Description
PRIVILEGE_NAME	VARCHAR(20)	Indicates the name of privilege as reflected in the following list: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW CREATE CACHE ALTER CACHE CREATE TABLESPACE If the value of DATA_SERVICE_NAME is _SERVER_, which corresponds to the SAS® Federation Server, the ADMINISTER and TRACE privileges can also be displayed. The ADMINISTER and TRACE privileges can be set on the SAS® Federation Server only.
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type can be GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege is grantable. The only valid value is N (No).

DSN_CONTENT View

The DSN_CONTENT view contains one or more rows per configured DSN. Each row contains a portion of the DSN content, ordered by sequence. If DATA_SOURCE_NAMES.format is STANDARD, then the content column contains driver connection string syntax.

The table below lists the columns that are associated with the DSN_CONTENT view:

Name	Type	Description
DSN_NAME	VARCHAR(256)	Specifies the unique name of the DSN.
SEQUENCE	INTEGER	Specifies the configuration chunk sequence.
CONTENT	VARCHAR(1024)	Specifies the DSN content.

DSN_LINEAGE View

The DSN_LINEAGE view contains information for federated DSNs. There is one entry per referenced DSN , so a federated DSN containing a reference to two DSNs would have two entries in this view.

The table below lists the columns that are associated with the DSN_LINEAGE view:

Name	Type	Description
DSN_NAME	VARCHAR(256)	Specifies the unique name of the DSN.
CHILD_DSN_NAME	VARCHAR(256)	Specifies the unique name of the child DSN.

DSN_PRIVILEGES and EFFECTIVE_DSN_PRIVILEGES Views

Displays privileges for users and groups on each data source name (DSN) and indicates inheritance. Both views show all direct (explicit) and inherited privileges based on the privileges of the user and group, or its group membership.

The DSN_PRIVILEGES result set contains rows for users and groups that have the CONNECT privilege explicitly set on either the server, service, or DSN. If a user or group does not have any direct privilege, it will not be shown in this view. It is a condensed view of the EFFECTIVE_DSN_PRIVILEGES view.

The EFFECTIVE_DSN_PRIVILEGES result set contains rows for all users and groups that have any privilege directly set or a privilege can be derived from its group membership. For example, if a user does not have any privileges set on any of the SAS® Federation Server objects, the user will still be in the result set if the user is a member of a group that has a direct privilege set.

Note: Both of these views can return very large result sets depending on the configuration of SAS® Federation Server. Subsetting on DATA_SERVICE, CATALOG_NAME, and/or SCHEMA_NAME can reduce the size of the result set.

Name	Type	Description
DSN_NAME	VARCHAR(256)	Specifies the unique name of the DSN.
DATA_SERVICE	VARCHAR(256)	Specifies the name of the data service.
GRANTOR_ID	VARCHAR(256)	Specifies the AuthID of the user that granted or denied the privilege.
GRANTOR	VARCHAR(256)	Specifies the name of the user who is granted or denied the privilege.

Name	Type	Description
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE_ID	VARCHAR(256)	Specifies the AuthID of the user that is granted or denied the privilege.
GRANTEE	VARCHAR(256)	Specifies the name of the user who is granted or denied the privilege.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).
PRIVILEGE_NAME	VARCHAR(20)	Specifies the privilege name: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the user can grant this privilege to other users. The only valid value is N (No).
INHERITED	CHAR(1)	Indicates if the privilege is inherited as either Y or N.
SOURCE_OBJECT_LEVEL	INTEGER	Specifies the object level where the privilege is inherited, as one of the following values: 0 — Server 1 — Data service 2 — DSN
SOURCE_GRANTEE_ID	VARCHAR(256)	Specifies the AuthID of a group or user from which the privilege is derived.
SOURCE_GRANTEE	VARCHAR(256)	Specifies the name of the group or user from which the privilege is derived.
SOURCE_GRANTEE_TYPE	CHAR(1)	Specifies the source_grantee type U (User) or G (Group).

IDENTITY View

The IDENTITY view returns identity information for a user connected to SAS Federation Server. The table below lists the columns that are associated with this view:

Name	Type	Description
USER_NAME	VARCHAR(256)	Specifies the name of the user that was created with the Authentication Server.
AUTH_DOMAIN	VARCHAR(256)	Specifies the domain name of the authenticated user. For example, if you connect to SAS Federation Server with the **`local\myuser`** account, the auth_domain is local.
AUTH_ID	VARCHAR(256)	Specifies the user name for the authenticated user. For example, if you connect to SAS Federation Server with the **`local\myuser`** account, the auth_id is myuser.

MESSAGES View

The MESSAGES view contains one or more messages associated with an operation. Each MESSAGE_ID represents one or more messages. Each message in a MESSAGE_ID has a unique number MESSAGE_NUM, and is separated into pieces that will fit within DATA. The SEQUENCE column can be used to order the entries within a MESSAGE_ID, and MESSAGE_NUM identifies individual messages.

Note: Rows are visible only if the rows referenced by the message are visible to the user.

Name	Type	Description
DATA	WVARCHAR(128)	Specifies a piece of the message text. Not NULL.
MESSAGE_ID	INTEGER	The unique ID of the messages stored in the table. Not NULL.
SEQUENCE	INTEGER	Orders the message entries for the entire MESSAGE_ID. Not NULL.
MESSAGE_NUM	INTEGER	Used to indicate separate messages within a MESSAGE_ID. Not NULL.

OBJECTS View

The OBJECTS view displays the name and system identifier for each schema object. This view is visible to SAS Federation Server System Users and Administrators. The table below lists the columns that are associated with this view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the catalog name.
SCHEMA_NAME	VARCHAR(256)	Specifies the schema name.
OBJECT_NAME	VARCHAR(256)	Specifies the object name.
OBJECT_TYPE	INTEGER	Object type: 1 – Table 2 – SAS View 3 – Cache 4 – SAS Package 5 – Relation
FLAGS	INTEGER	Flags 1 - Definer's rights. For FedSQL views, the definer's rights bit indicates the view executes under the privileges of the view's schema owner rather than the invoker. 2 - Object was implicitly added as a result of a GRANT (as opposed to explicitly added through a user DDL). The object can be automatically removed if the grant is revoked.

OBJECT_PRIVILEGES View

The OBJECT_PRIVILEGES view displays the privilege descriptors for all object-level privileges. All privileges for a single table can be obtained by concatenating rows matching the correct priv_id and ordering the results by sequence. The table below lists the columns that are associated with the OBJECT_PRIVILEGES view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog.
SCHEMA_NAME	VARCHAR(256)	Specifies the name of the schema.
OBJECT_NAME	VARCHAR(256)	Specifies the name of the object.

Name	Type	Description
GRANTOR	VARCHAR(256)	Specifies the user name of the grantor.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE	VARCHAR(256)	Specifies the name of the user who is granted privileges.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).
PRIVILEGE_NAME	VARCHAR(20)	Specifies the privilege name as one of the following values: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW CREATE CACHE ALTER CACHE
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege is grantable. The only valid value is N (No).
PREDICATE	VARCHAR(128)	Portion of the row-level security (RLS) predicate. The predicate might spawn multiple rows; is nullable.
PREDICATE_SEQUENCE	SMALLINT	specifies the sequence number of the portion of the RLS predicate; is nullable.
PRIV_ID	BIGINT	Privilege identifier.

PRIVILEGES and EFFECTIVE_PRIVILEGES Views

Displays the privileges, including inheritance, for users and groups on schemas, catalogs and data services. Both views show all direct (explicit) and inherited privileges based on the privileges of the user and group, or its group membership.

The PRIVILEGES result set contains rows for users and groups that have any privilege directly set. If a user or group does not have any direct privilege, it will not be shown in this view. It is a condensed view of the EFFECTIVE_PRIVILEGES view.

The EFFECTIVE_PRIVILEGES result set contains rows for all users and groups that have any privilege directly set or a privilege can be derived from its group membership. For example, if a user does not have any privileges set on any of the Federation Server objects, the user will still be in the result set if the user is a member of a group that has a direct privilege set.

By default, if a privilege is not explicitly listed in the result sets, it is denied.

Note: Both of these views can return very large result sets depending on the configuration of SAS Federation Server. Subsetting on DATA_SERVICE, CATALOG_NAME, and/or SCHEMA_NAME can reduce the size of the result set.

The table below lists the columns that are associated with the PRIVILEGES and EFFECTIVE_PRIVILEGES view:

Name	Type	Description
DATA_SERVICE	VARCHAR(256)	Specifies the name of the data service.
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog.
SCHEMA_NAME	VARCHAR(256)	Specifies the name of the schema.
GRANTOR_ID	VARCHAR(256)	Specifies the AuthID of the user that granted or denied the privilege.
GRANTOR	VARCHAR(256)	Specifies the name of the user who is granted or denied the privilege.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE_ID	VARCHAR(256)	Specifies the AuthID of the user that is granted or denied the privilege.
GRANTEE	VARCHAR(256)	Specifies the name of the user who is granted or denied the privilege.
PRIVILEGE_NAME	VARCHAR(20)	Specifies the privilege name as one of the following values: SELECT UPDATE INSERT REFERENCES
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege can be granted. The only valid value is or N (No).
INHERITED	CHAR(1)	Indicates if the privilege is inherited as either Y or N.

Name	Type	Description
SOURCE_OBJECT_LEVEL	INTEGER	Specifies the object level where the privilege is inherited, as one of the following values: 0 — server 1 — data service 2 — catalog 3 — schema
SOURCE_GRANTEE_ID	VARCHAR(256)	Specifies the AuthID of a group or user from which the privilege is derived.
SOURCE_GRANTEE	VARCHAR(256)	Specifies the name of the group or user from which the privilege is derived.
SOURCE_GRANTEE_TYPE	CHAR(1)	Specifies the source_grantee type as U (User) or G (Group).

SCHEMAS View

The SCHEMAS view contains the name and system identifier for each schema. The table below lists the columns that are associated with the SCHEMAS view:

Name	Type	Description
SCHEMA_NAME	VARCHAR(256)	Specifies the schema name.
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog. If not applicable, the value is NULL.
USER_NAME	VARCHAR(256)	Specifies the user name of the schema owner.

SCHEMA_PRIVILEGES View

The SCHEMA_PRIVILEGES view displays the privilege descriptors for all schema-level privileges. The table below lists the columns that are associated with the SCHEMA_PRIVILEGES view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog. If not applicable, the value is NULL.
SCHEMA_NAME	VARCHAR(256)	Specifies the name of the schema.

Name	Type	Description
GRANTOR	VARCHAR(256)	Specifies the user name of the grantor.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE	VARCHAR(256)	Specifies the name of the user who is granted privileges.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).
PRIVILEGE_NAME	VARCHAR(20)	Specifies the privilege name as one of the following values: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLESPACE CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW CREATE CACHE ALTER CACHE CREATE TABLESPACE
PRIVILEGE_TYPE	VARCHAR(5)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege is grantable. The only valid value is N (No).

X_COLUMN_PRIVILEGES/ X_EFFECTIVE_COLUMN_PRIVILEGES View

The X_COLUMN_PRIVILEGES and the X_EFFECTIVE_COLUMN_PRIVILEGES views contain both the privileges for users and groups on all objects[2], and indicate inheritance. They show all direct (explicit) and inherited privileges based on the privileges of the user or group or its group membership. Unlike most other views, the views do not strictly derive the information from system tables. It will merge metadata from the physical data sources with metadata in system tables to produce a complete result set for all objects.

The X_COLUMN_PRIVILEGES result set contains rows for users and groups that have any privilege directly set. If a user or group does not have any direct privilege, it will not

be shown in this view. It is a condensed view of the X_EFFECTIVE_COLUMN_PRIVILEGES view.

The X_EFFECTIVE_COLUMN_PRIVILEGES result set contains rows for all users and groups that have any privilege directly set or a privilege can be derived from its group membership. For example, even if a user does not have any privileges directly set, records for this user will be in the result set if any of the groups in its group hierarchy has a privilege directly set.

Note: Both of these views can return very large result sets depending on the configuration of Federation Server. Subsetting on DATA_SERVICE, CATALOG_NAME, and/or SCHEMA_NAME can reduce the size of the result set.

Name	Type	Description
DATA_SERVICE	VARCHAR(256)	Specifies the data service name.
CATALOG_NAME	VARCHAR(256)	Specifies the catalog name.
COLUMN_NAME	VARCHAR(256)	Specifies the column name.
SCHEMA_NAME	VARCHAR(256)	Specifies the schema name.
GRANTOR_ID	VARCHAR(256)	Specifies the AuthID of the grantor.
GRANTOR	VARCHAR(256)	Specifies the name of the grantor. This field could be NULL if the user no longer exists.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as R (Role) or U (User).
GRANTEE_ID	VARCHAR(256)	Specifies the AuthID of the grantee.
GRANTEE	VARCHAR(256)	Specifies the name of the grantee.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).
PRIVILEGE_NAME	VARCHAR(256)	Name of privilege as reflected in the following list: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW
PRIVILEGE_TYPE	VARCHAR(256)	Specifies the privilege type as GRANT or DENY.

Name	Type	Description
GRANTABLE	CHAR(1)	Specifies if the privilege is grantable. The only valid value is N (No).
INHERITED	CHAR(1)	Specifies if the privilege is inherited with Y (Yes) or N (No).
SOURCE_OBJECT_LEVEL	INTEGER	Specifies the object level where the privilege is inherited from: 0 – server 1 – data service 2 – catalog 3 – schema 4 – object 5 - column
SOURCE_GRANTEE_ID	VARCHAR(256)	AuthID of group or user the privilege is derived from.
SOURCE_GRANTEE	VARCHAR(256)	Specifies the group or user name the privilege is derived from.
SOURCE_GRANTEE_TYPE	CHAR(1)	Specifies the grantee as U (User) or G (Group).
OBJECT_NAME	VARCHAR(256)	Specifies the name of the object.

[2] Current list of objects includes:

- table server
- data services
- catalogs
- schemas

X_OBJECT_PRIVILEGES/ X_EFFECTIVE_OBJECT_PRIVILEGES View

The X_OBJECT_PRIVILEGES and the X_EFFECTIVE_OBJECT_PRIVILEGES views contain both the privileges for users and groups on all objects[2] and indicates inheritance. They show all direct (explicit) and inherited privileges based on the privileges of the user and group or its group membership. Unlike most other views, the views do not strictly derive the information from system tables. It will merge metadata from the physical data sources with metadata in system tables to produce a complete result set for all objects.

The X_OBJECT_PRIVILEGES result set contains rows for users and groups that have any privilege directly set. If a user or group does not have any direct privilege, it will not be shown in this view. It is a condensed view of the X_EFFECTIVE_OBJECT_PRIVILEGES view.

The X_EFFECTIVE_OBJECT_PRIVILEGES result set contains rows for all users and groups that have any privilege directly set or a privilege can be derived from its group membership. For instance, even if a user does not have any privileges directly set, records for this user will be in the result set if any of the groups in its group hierarchy has a privilege directly set.

If a privilege is not explicitly listed in the result sets, it is DENIED by default.

Name	Type	Description
DATA_SERVICE	VARCHAR(256)	Specifies the data service name.
CATALOG_NAME	VARCHAR(256)	Specifies the catalog name.
SCHEMA_NAME	VARCHAR(256)	Specifies the schema name.
GRANTOR_ID	VARCHAR(256)	Specifies the AuthID of the grantor.
GRANTOR	VARCHAR(256)	Specifies the name of the grantor. This field could be NULL if the user no longer exists.
GRANTOR_TYPE	CHAR(1)	Specifies the grantor type as U (User) or R (Role).
GRANTEE_ID	VARCHAR(256)	Specifies the AuthID of the grantee.
GRANTEE	VARCHAR(256	Specifies the grantee name.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).
PRIVILEGE_NAME	VARCHAR(256)	Name of privilege as reflected in the following list: SELECT UPDATE INSERT DELETE EXECUTE REFERENCES CREATE TABLE ALTER TABLE DROP TABLE CREATE VIEW ALTER VIEW DROP VIEW CREATE CACHE ALTER CACHE
PRIVILEGE_TYPE	VARCHAR(256)	Specifies the privilege type as GRANT or DENY.
GRANTABLE	CHAR(1)	Specifies if the privilege is grantable. The only valid value is N (No).

Name	Type	Description
INHERITED	CHAR(1)	Specifies if the privilege is inherited with Y or N.
SOURCE_OBJECT_LEVEL	INTEGER	Specifies the object level where the privilege is inherited from: 0 – server 1 – data service 2 – catalog 3 – schema 4 – object 5 - column
SOURCE_GRANTEE_ID	VARCHAR(256)	AuthID of group or user the privilege is derived from.
SOURCE_GRANTEE	VARCHAR(256)	Specifies the group or user name the privilege is derived from.
SOURCE_GRANTEE_TYPE	CHAR(1)	Specifies the grantee as U - User or G - Group.
OBJECT_NAME	VARCHAR(256)	Specifies the name of the object.

[2] Current list of objects includes:

- server
- data services
- catalogs
- schemas

Glossary

ACID
See atomicity, consistency, isolation, durability

American National Standards Institute
the organization that coordinates the development of voluntary consensus standards for products, services, processes, systems, and personnel in the United States. ANSI works with the International Organization for Standardization to establish global standards. Short form: ANSI.

ANSI
See American National Standards Institute

API
See application programming interface

application programming interface
a set of software functions that facilitate communication between applications and other types of programs or services. Short form: API.

Application Response Measurement
the name of an application programming interface that was developed by an industry partnership and which is used to monitor the availability and performance of software applications. ARM monitors the application tasks that are important to a particular business. Short form: ARM.

ARM
See Application Response Measurement

atomicity, consistency, isolation, durability
the characteristics of a transaction, such as a group of SQL statements, in an RDBMS that support commit and rollback operations. The characteristics include atomicity (the execution of a transaction, either committed or rolled back); consistency (the successful application of the consistency rules of an RDBMS that commits only valid data to the database); isolation (the separation of a transaction from all other concurrent processes in an RDBMS); and durability (the persistence or repeatability of a transaction under all conditions, including system failure, after which a transaction can be re-created from a log that contains committed transactions). Short form: ACID

authentication
See client authentication

authorization
the process of determining the permissions that particular users have for particular resources. Authorization either permits or denies a specific action on a specific resource, based on the user's identity and on group memberships.

client authentication
the process of verifying the identity of a person or process for security purposes.

connection string
information that defines how to connect an application to the data. In SAS Federation Server, a connection string identifies the query language syntax that the application submits, as well as the information that is required to connect to a data source or data sources.

data source name
a persistent identifier that is associated with a data source definition. The data source definition specifies how to locate and access a data source, including any authentication (such as a user name and password) that a user must provide. Short form: DSN.

data type
an attribute of every column in a table or database, indicating the type of data in the column and how much physical storage it occupies.

definer's rights view
a view that is created by a schema owner. Definer's rights views are required for data caching in SAS Federation Server.

driver
a special-purpose software program that enables two disparate software programs, such as an application and an API, to interact.

DSN
See data source name

encryption
the act or process of converting data to a form that is unintelligible except to the intended recipients.

federated DSN
a data source name that references multiple data sources. The data sources can be on the same DBMS, or on a different one.

grouping data source name
See federated DSN

grouping DSN
See federated DSN

Integrated Object Model
the set of distributed object interfaces that make SAS software features available to client applications when SAS is executed as an object server. Short form: IOM.

invoker's rights view
a federated view or cache that is accessed using the current user's authorization instead of the schema owner's authorization. See also "definer's rights view."

IOM
See Integrated Object Model

Java Virtual Machine
a software application that can execute Java bytecode, on either a client or a server, enabling Java programs to be run on many different hardware and software platforms. Short form: JVM.

join
an operation that combines data from two or more tables. A join is typically created by means of SQL (Structured Query Language) code or a user interface.

JVM
See Java Virtual Machine

MDS
See Memory Data Store

Memory Data Store
a transactional data cache that runs strictly in-memory. Because there is no back up data storage, changes are lost when the in-memory database is closed.

result set
the set of rows or records that a server or other application returns in response to a query.

RLS
See row-level security

RLS predicate
See row-level security predicate

row-level security
a security feature that controls access to rows and columns in a table in order to prevent users from accessing restricted data.

row-level security predicate
a query that restricts the rows that are available to grantees for specified operations. Only rows that match the predicate can be accessed by the grantees.

scrollable cursor
a device that enables an application to set a position on any row in a result set. For example, a scrollable cursor can back up and revisit a row, start at the end of the file and work backward, skip some rows, or go directly to a specific row.

serializability
a capability commonly required in database processing that ensures the highest level of isolation between transactions for the purposes of concurrency control.

SQL
See Structured Query Language

Structured Query Language
a standardized, high-level query language that is used in relational database management systems to create and manipulate objects in a database management system. SAS implements SQL through the SQL procedure. Short form: SQL.

thread
the smallest unit of processing that can be scheduled by an operating system.

threaded processing
processing that is performed in multiple threads in order to improve the speed of CPU-bound applications.

time-out
an error condition that is produced when a required response from a device or program is not received after a specified length of time.

transactional data store
a storage mechanism for transactional data that is characterized by ACID features (atomicity, consistency, isolation and durability).

type
See data type

Unicode
a 16-bit encoding that is the industry standard for supporting the interchange, processing, and display of characters and symbols from most of the world's writing systems.