SAS® Federation Server 4.2: Administrator's Guide, Second Edition

Contents

What's New in SAS Federation Server

Overview

The following new features have been added to SAS Federation Server 4.2:

- SAS Federation Server connectivity to SAS Scalable Performance Data Server (SPDS) allowing READ, WRITE, and UPDATE to SPDS tables.

- replacement of DataFlux Authentication Server by SAS Metadata Server for authentication and other permission-based functions

- support for SAS DS2 model scoring code in your database

- enhanced data masking and encryption support

- cache enhancements that include cache refresh for data held in memory (MDS)

- embedded data quality and cleansing functions in data views

- Read/Write access to Hadoop (HIVE) using the SAS Federation Server Driver for Apache Hive

- new SAS Federation Server driver that allows shared data sources across multiple SAS Federation Servers

SAS Federation Server Driver for SPD Server

Tables located on SAS SPD Server can be accessed for reading, writing, and update by SAS Federation Server with the SAS Federation Server Driver for SPD Server (Driver for SPD Server). The Driver for SPD Server provides connectivity from a SAS Federation Server on any host to an SPD server running anywhere. The Driver for SPD Server integrates with UNIX hosts as well as Windows. See "SAS Federation Server Driver for SAS Scalable Performance Data (SPD) Server" .

SAS Metadata Server

SAS Metadata Server replaces DataFlux Authentication Server as the authentication provider. SAS Metadata Server provides access for user and group objects, as well as other permission-based functions such as shared logins and trusted users. SAS

Federation Server objects are populated from objects created in SAS Metadata Server. There is no longer a need to create federation server objects in SAS Federation Server Manager. See the *SAS Federation Server: Administrator's Guide* for information about how SAS Federation Server works with SAS Metadata Server.

SAS DS2 Language Support

DS2 is a SAS proprietary programming language that is used for advanced data manipulation. DS2 provides capabilities not available through SQL, such as scoring models. You can also use DS2 code to run data quality functions using SAS Federation Server Manager.

New Data Masking Rules

New data masking features include TRANC, which transliterates characters from the input string to characters in the output string. A series of random data masking rules are also available. See the "Data Masking Rule Types and Arguments" for information about each of these new rules and their functionality in SAS Federation Server Manager.

Enhanced Cache Operations

Federation Server now has the capability of refreshing cached data in MDS after a server restart. In previous releases, cached data that was held in memory was deleted if the server was restarted or shut down. Now the cache held in memory is retained and refreshed at server restart by default. See the SET RESTART='REFRESH' option in the CREATE CACHE statement for details. Also see "Data Caching" for additional information about creating a cache table.

Data Quality and Cleansing

The new data quality and cleansing is implemented using SAS Quality Knowledge Base (QKB) with FedSQL and DS2. The data quality methods use data quality rules from the SAS QKB in order to cleanse data. The SAS Quality Knowledge Base (QKB) is a collection of files that store data and logic that define data management operations such as parsing, standardization, and matching. SAS software products refer to the QKB when performing data management operations, also referred to as data cleansing, on your data. See "Data Quality for SAS Federation Server" for additional information.

SAS Federation Server Driver for Apache Hive

SAS Federation Server supports big data with the SAS Federation Server Driver for Apache Hive (Driver for Hive). The Driver for Hive enables SAS Federation Server to query and manage large data sets that reside in distributed storage. The Driver for Hive supports multiple versions of Hadoop. See "About the SAS Federation Server Driver for Apache Hive". for additional information including configuration and connection options.

SAS Federation Server Driver

You can now share data sources across multiple SAS Federation Servers with the new federation server driver (FEDSVR). The driver enables you to define a connection from one SAS Federation Server to another SAS Federation Server. This connectivity can be useful for distributing work load or to federate data between various federation servers. See "About the Federation Server Driver" for additional information including configuration and connection options.

Chapter 1
Overview

Introduction

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 sources. 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 IBM DB2®, Netezza, Oracle®, SAP®, SAP Hana®, Microsoft® SQL Server®, PostgreSQL®, Teradata®, and Greenplum®.

- Access to distributed storage with the SAS Federation Driver for Apache Hive™.

- 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. SAS Federation Server also includes its own SQL syntax, Federated Query Language (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 source access control that includes user permissions and row-level security.

The following figure illustrates the architecture of SAS Federation Server:

Figure 1.1 *SAS Federation Server Architecture*
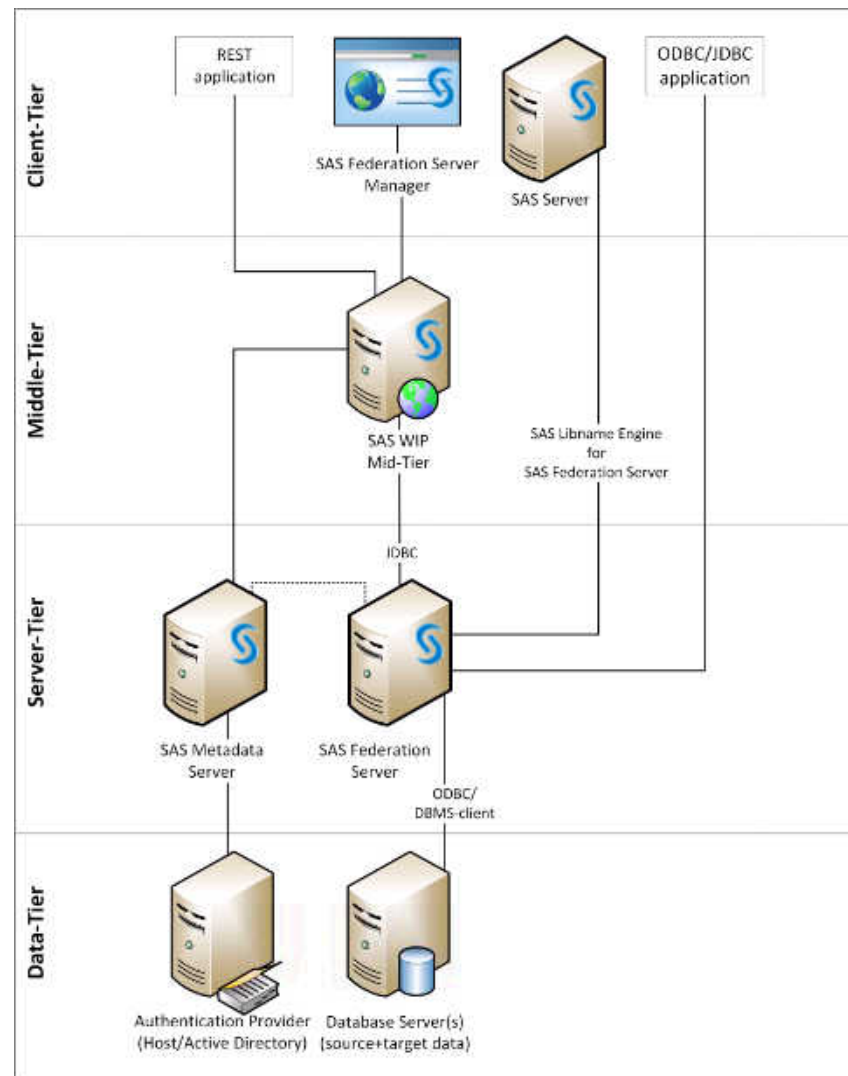
Services Provided by SAS Federation Server

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 enables you to work with multiple data sources as if they were a single resource, regardless of where the information is stored. SAS Federation Server supports SAS data sets, SAP, distributed storage with Apache Hive, and third-party relational databases that include:

- IBM DB2, Netezza

- Oracle

- Microsoft SQL Server

- Teradata

- Greenplum

- PostgreSQL

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 source.

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 6.

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.

See SAS Federation Server Security for information about these security features.

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. SAS Federation Server provides several types of drivers that you use to connect to the server. The following topics describe the drivers that are available with SAS Federation Server.

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 SAS Federation Server Driver for SAP, and to process an Oracle table, an application uses the SAS Federation Server 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 the supported data sources and their connection options, see the SAS Federation Server Driver Reference .

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. SAS Federation Server supports FedSQL and DS2 languages.

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. 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, Apache Hive, and several third-party relational databases.

The following table lists data sources with their data service names:

Data Sources	Data Service Name	Default Drivers	Supported Drivers
SAS data sets	BASE	BASE	BASE
SASHDAT-Hadoop on Grid	SASHDAT	SASHDAT	SASHDAT
Distributed storage, Apache Hive	HIVE	HIVE	ODBC, HIVE
DB2	DB2UNXPC	DB2	ODBC, DB2
Greenplum	GREENPLUM	Greenplum	ODBC, Greenplum
Netezza	NETEZZA	Netezza	ODBC, Netezza
Oracle	ORACLE	ORACLE	ODBC, ORACLE
PostgreSQL	POSTGRESQL	PostgreSQL	ODBC, PostgreSQL
SAP	SAP	SAP	SAP
SAP HANA	SAPHANA	SAP HANA	ODBC, SAP HANA
SQL Server	SQLSERVER or SQLSVR	ODBC	ODBC
Teradata	TERADATA	TERADATA	ODBC, TERADATA
Native Catalogs ODBC to data sources that support native catalogs.	ODBC_FED	ODBC	ODBC
Logical Catalogs ODBC to data sources that don't support native catalogs. In this case, SAS Federation Server provides a logical catalog.	ODBC	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.

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 Federation Server Driver for Base SAS" .

SAP, SAP HANA

SAS Federation Server works with SAP and SAP HANA. For SAP installation and configuration, including connection options, see "SAS Federation Server Driver for SAP " on page 198 . Connection options for SAP HANA are outlined in the "SAS Federation Server Driver for SAP HANA "..

Apache Hive

Using the SAS Federation Server Driver for Apache Hive, SAS Federation Server can query and manage large data sets that reside in distributed storage. See the "SAS Federation Server Driver for Apache Hive" for connection information.

Third-Party Relational Databases

SAS Federation Server can access data in several third-party relational databases. 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:

- "SAS Federation Server Driver for DB2"
- "SAS Federation Server Driver for Greenplum"
- "SAS Federation Server Driver for Netezza"
- "SAS Federation Server Driver for ODBC"
- "SAS Federation Server Driver for Oracle"
- "SAS Federation Driver for PostgreSQL"
- "SAS Federation Server Driver for Teradata"

Chapter 2

Getting Started with SAS Federation Server

Introducing SAS Metadata Server

About SAS Metadata Server

Administration
Administration for SAS Metadata Server is performed using SAS Management Console.
While the SAS Metadata Server is running and online, you can use SAS Management
Console to connect to the SAS Metadata Server and view and manage the objects that
are stored in the server's metadata repositories.

Server Backups

SAS Metadata Server is configured to perform unassisted server backups every day of the week except on Sunday. On Sunday, backup is performed for all servers in the deployment. Backups are retained for seven days in the following directory: `SAS-configuration-directory/Lev1/SASMeta/MetadataServer/Backups`. The metadata server is configured to send an alert email message to the assigned administrator in the event of a system, backup, or recovery failure. To learn about the metadata server backup facility, see "Backing Up the SAS Metadata Server" in the *SAS Intelligence Platform: System Administration Guide*.

SAS Federation Server Functionality as Metadata

SAS Metadata Server replaces DataFlux Authentication Server as the authentication provider for SAS Federation Server. Additionally, some of the objects and configurations that were created in DataFlux Authentication Server or SAS Federation Server are now created in SAS Metadata Server:

- **System User**: A **SAS Federation Server System User** is created at installation as **sasfedadm**. This account is a member of the **SAS Federation Server Administrators** group.

- **Trusted User**: Referred to as 'the SAS Trusted User', the **sastrust** account is created during installation or upgrade. This account replaces the trusted user that was created in DataFlux Authentication Server.

- **Administrator**: SAS Federation Server administrator accounts are facilitated in SAS Metadata Server by granting membership to the **SAS Federation Server Administrators** group, for selected accounts.

- **Shared login**: The shared login account that was formerly created in DataFlux Authentication Server, is now created as a shared login group in SAS Metadata Server.

- **Federation server object**: A federation server object is created during installation and appears as a server object in SAS Management Console. The federation server object is no longer defined in SAS Federation Server Manager.

See the *SAS Management Console: Guide to Users and Permissions* for information about creating users and groups for SAS Metadata Server: http://support.sas.com/documentation/cdl/en/mcsecug.

Additional configuration options were added to facilitate the use of SAS Metadata Server in the SAS Federation Server environment:

- The use of **metaprofile** to configure the SAS Federation Server environment. This replaces the port and host configuration used in previous versions of SAS Federation Server.

- Metauser and Metapass to specify credentials to connect to SAS Federation Server.

See the Configuration Reference for additional information about these configuration options.

Post-Installation Configuration

Overview

After you install SAS Federation Server, you might need to perform additional configuration steps before you can use SAS Federation Server. At the end of the installation, the SAS Deployment Wizard produces an HTML document named Instructions.html. If your server tier and middle tier are hosted on separate machines, there is an Instructions.html file for each machine. The Instructions.html file is located in `SAS\Config\Lev#\Documents\`. Here is an outline of tasks that require attention:

1. Verify that all installation and configuration steps in the Instructions.html file have been completed.

2. Create users, groups, and roles.

3. (Optional) Specify an encryption level for SAS Federation Server.

SAS Metadata Server

User Requirements and Roles

To access SAS Federation Server Manager, users might require group membership that includes assignment of specific roles.

- A non-administrator user requires the **Federation Server Manager: Operation** role.

- An administrator object requires membership to the **SAS Federation Server Administrators** group, with the **ManageMemberMetadata** permission. The **Federation Server Manager: Operation** role is assigned by default.

- The **SAS Federation Server System User** account that is created at installation is **sasfedadm**. This account is a member of the **SAS Federation Server Administrators** group.

SAS Federation Server Administrators

A user becomes an administrator when their account is added to the Federation Server Administrators group in SAS Metadata Server. This action grants the ADMINISTER privilege to the user object. Only the SAS Federation Server System user, sasfedadm, can perform this action, as well as the SAS Metadata Administrator (sasadm).

Only the system user has the authority to grant or revoke the ADMINISTER privilege through the use of administration DDL. The ADMINISTER permission is available on the server object only.

Specify Server Encryption Level

Use the following procedure to specify or change the encryption level for a particular SAS Federation Server.

1. Using SAS Management Console, locate your federation server object by expanding **Environment Manager** ⇨ **Server Manager** ⇨ **Federation Server -** *hostname* - **logical server**.

2. Expand the logical server entry and select the server definition that you want to change encryption for. The **Connections** tab displays the current connections defined for the selected server.

3. On the **Connections** tab, select a **connection** and right-click. Select **Properties** from the drop-down menu.

4. Select the **Options** tab and select **Advanced Options**.

5. Select the **Encryption** tab and select an option from the **Server encryption algorithm** list menu.

6. Click **OK** to exit the Advanced Options dialog box, and click **OK** to close connection properties.

7. Restart SAS Federation Server to update the server encryption algorithm.

Shared Login Accounts

About Shared Logins

Shared logins consist of a shared login key, the login account, and the users or groups who are members of the (shared) login account. The SAS Federation Server administrator creates and controls the shared logins for SAS Federation Server.

When using a shared login to authenticate to a data source, users do not need to know the credentials of the shared login. The shared login retrieves credentials for the user who is logged on and provides the credentials to SAS Federation Server. In turn, the server connects the user to the database through the appropriate data service or data source name (DSN).

Outline of Shared Login Tasks

The implementation of shared logins has changed in SAS Federation Server 4.2. Here is a summary of the tasks:

• Create a shared login key for SAS Federation Server using administrative DDL or in SAS Federation Server Manager in the properties of a federation server object. The shared login key is case sensitive. The key that is defined in SAS Federation Server must match the key that is part of the shared login definition in the SAS Metadata Server.

• Create a shared login account (group) in SAS Metadata Server using SAS Management Console. The shared login account includes the login to be shared and its domain.

• Add consumers of the shared login as members of the shared login account. Consumers are SAS Federation Server user accounts or groups. You should never use the actual shared login group as a consumer group in a DSN.

• Create a data service for the applicable data source. In the DSN, specify that the data will be accessed with a shared login.

About the Authentication Domain

When establishing connection to the SAS Federation Server, the following logic is used to find the proper login:

• If connecting with a DSN configured to use a personal or group login, SAS Federation Server uses the authentication domain associated with the data service to look up a login for the user.

- If connecting with a DSN configured to use a shared login, SAS Federation Server uses the authentication domain associated with the data service and appends the domain with a suffix of "**@<shared login key>**" to look up a login for the user.

Creating a Shared Login

The tasks presented in the following topics outline the basic steps to create a shared login for SAS Federation Server:

1. Set a shared login key (SAS Federation Server Manager).

2. Create the shared login account (SAS Management Console).

3. Create a data service and DSN for the data source (SAS Federation Server Manager).

Set a Shared Login Key

The shared login key is used when configuring an authentication domain in SAS Metadata Server. The shared login key is case sensitive. the following steps show how to set a shared login key with SAS Federation Server Manager:

1. Locate the federation server object in the tree, and log on to the server if prompted to do so.

2. Select **Action Menu** ⇨ **Properties** in the upper left corner.

3. Click the **Security** tab and enter the shared login key.

4. Click **OK** to exit the properties dialog box.

TIP You can also use administration DDL to set a shared login key: **ALTER SERVER {OPTIONS (SHAREDLOGINKEY name-of-key) }**

Create a Shared Login Account

The shared login account is actually a group that serves as the shared login account, so the name of the group should reflect that (reference step 4a below).

1. Log on to SAS Management Console

2. On the **Plug-ins** tab, select **User Manager**.

3. Right-click and select **New** ⇨ **Group**.

4. In the New Group Properties dialog box:

   a. On the **General** tab, enter a name for the shared login (for example, Oracle Shared Login for FedServer).

   b. On the **Members** tab, add users and groups who will use the shared login.

   c. On the **Accounts** tab, add the account and password.

   d. Select **New** for Authentication Domain.

      - Enter an Authentication Domain name using this format:

        <data_service_domain>@<shared_login_key>

        For example, if the domain for the data service is OracleAuth and the shared login key is **FSKey1**, then the shared login domain must be **OracleAuth@FSKey1**. The shared login key is case sensitive and must match the shared login key that was set in SAS Federation Server Manager.

      - Select **Outbound only** and Trusted only for the domain.

Figure 2.1 *New Authentication Domain Dialog Box*
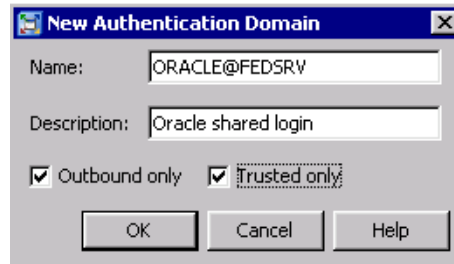
Outbound only: An outbound domain is used only to provide SAS applications with access to external resources, such as a third-party vendor database.

Trusted only: The trusted user is a privileged service identity that can act on behalf of all other users. A login in a trusted domain can be accessed only by a trusted user.

5. On the **Authorizations** tab, ensure that the SAS Administrators group has these permissions:

   • ManageMemberMetadata

   • ManageCredentialsMetadata

   • ReadMetadata

   • WriteMetadata

Add Members to the Shared Login Group

Once the shared login is configured, you must add users and groups as consumers of the shared login. Use the following procedure in SAS Management Console to add a user or group to a shared login.

1. On the **Plug-ins** tab, select **User Manager**.

2. Locate the shared login object, right-click, and select **Properties**.

3. In the Properties dialog box, on the **Members** tab, add users and groups who will use the shared login.

Figure 2.2 *Shared Login Consumer Membership*
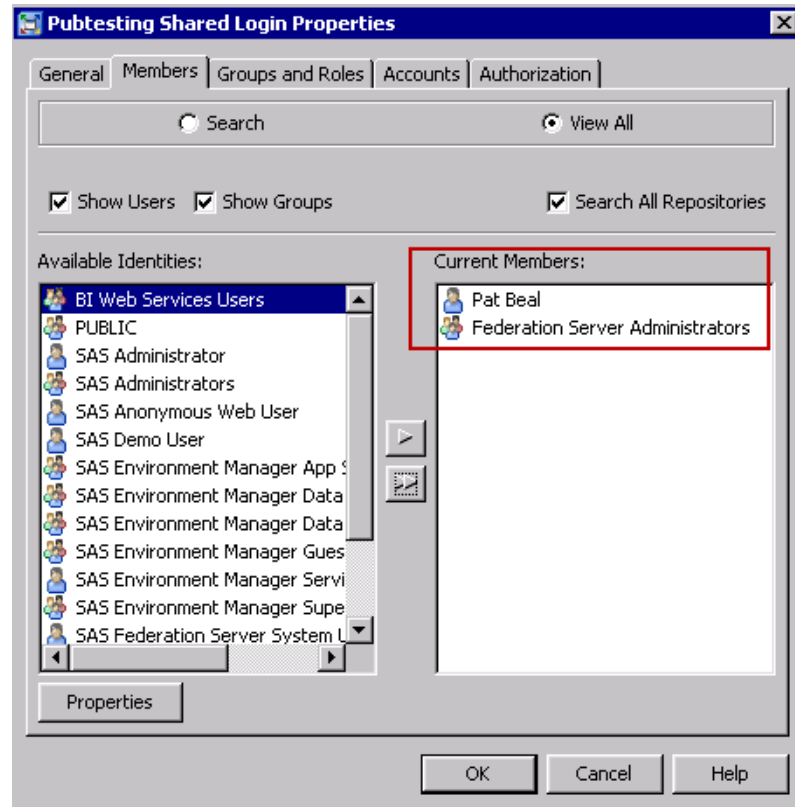
4. Click **OK** when you are finished.

Create a Data Service and DSN

When you create a data service, a DSN with the same name is automatically created for you. Use SAS Federation Server Manager to perform the following task.

1. Select a federation server object in the tree, and log on to the server if you are prompted

2. Select **Action** ⇨ **New Data Service**, or click the New Data Service icon ▤ on the toolbar.

3. In the Identification dialog box, enter the name of the data service and click **Next** to continue.

4. In the Authentication Domain dialog box, select an Authentication Domain from the list of available domains and click **Next** to continue.

Figure 2.3 Defining the Data Service Authentication Domain
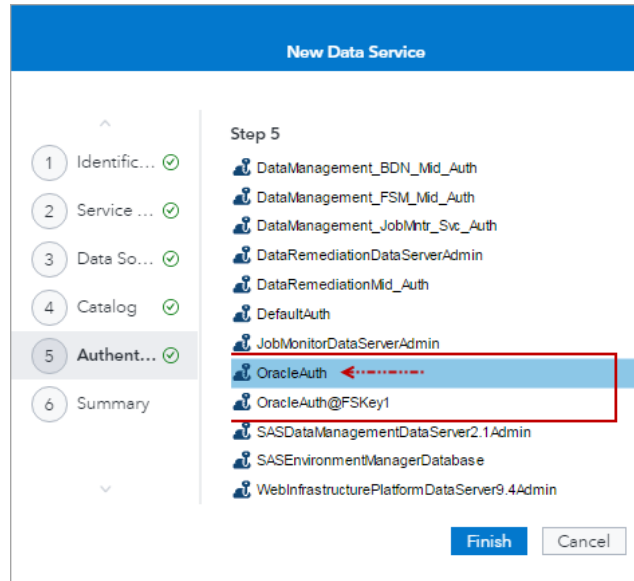
CAUTION:

Select a stand-alone data source domain. Do not select the domain with the shared login key that was created in SAS Metadata Server. When the DSN is set to use a shared login, SAS Federation Server appends the selected domain with @ and the shared login key and verifies that `data source@<shared login key>` exists in SAS Metadata as a valid authentication domain that includes user and password account information.

5. In the Summary dialog box, verify the settings and click **Finish**.

Set the Shared Login Indicator in the DSN

1. Select the **Data Source Names** tab affiliated with the Oracle data service that you just created. You should see a DSN that is named for the new data service.

2. Select the Action menu ⋮ , select **Properties**, and click **Next** until you reach the Access dialog box.

3. In the **Specify the type of login required to access this DSN** field, select the **Shared login** check box.

Figure 2.4 Shared Login Specification for DSN Access
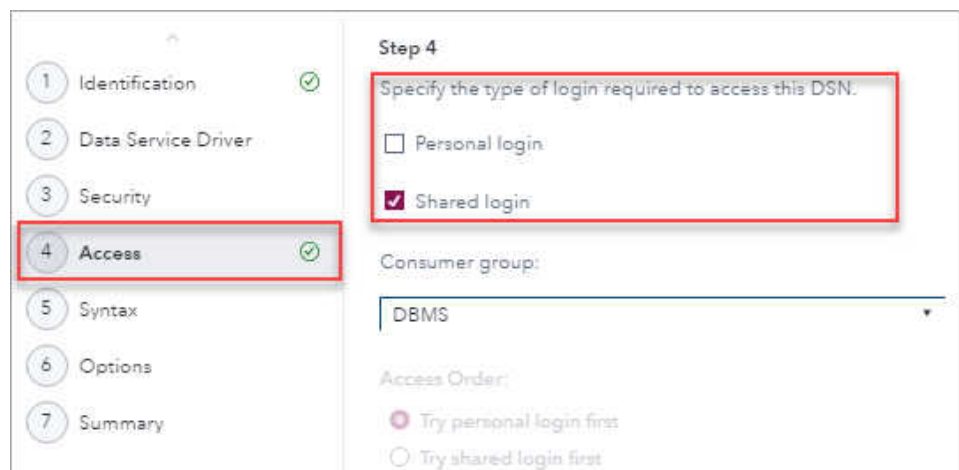
4. From the **Consumer group** drop-down list, select a group if necessary.

   Note: The Consumer group identifies which shared login should be used if a conflict occurs for a user. The Consumer group should be a group that is directly or indirectly a member of the shared login.

5. Click **Next**, **Next**, **Next**, and **Finish**.

ODBC Wire Protocol Branded Drivers

About the ODBC Wire Protocol Drivers

SAS Federation Server installs version 7.1 set of wire protocol ODBC drivers for several databases. The drivers are installed at **[*drive*]:/Program Files/DataFlux/ ODBC/7.1**. The database and database connection must also be configured as an ODBC data source.

Windows ODBC Configuration

To add an ODBC data source, use the ODBC Data Source Administrator in Microsoft Windows. Use the following procedure 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 **Drivers** tab to display the wire protocol drivers.

5. Select a driver and click **OK**.

6. In the ODBC Driver Setup dialog box, enter the Data Source Name, Description, and other configurations specific to your data source. These values are required, and can be obtained from your database administrator.

UNIX ODBC Configuration

SAS Federation Server includes an ODBC configuration tool, **dfdbconf**, that is used to configure the ODBC wire protocol drivers. The utility is located in the **/bin** directory of the SAS Federation Server installation path. The options are A –Add, D –Delete, and X – Exit.

To add an ODBC 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.

See "Configuring ODBC Connections" for additional configurations required for ODBC in an UNIX environment.

Using the SAS Federation Server Drivers

Before configuring the SAS Federation Drivers, you must set environment variables. See "Setting Environment Variables" for information to set environment variables for your particular data source.

When you are ready to configure your federation server driver, see the SAS Federation Server Driver Reference "Database Functionality and Driver Performance", which provides the connection options for your data source.

About the SAS Federation Server Accounts

Overview

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

- "The SAS Federation Server System User Account"

  The system user account is the most privileged account for SAS Federation Server. User account **sasfedadm** on SAS Metadata Server.

- "The Administrator Account and Federation Server Administrators Group"

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

- "The SAS Trusted User Account"

  The SAS Trusted User account establishes a trust relationship between the SAS Federation Server and the SAS Metadata Server. The trusted user account is used to retrieve shared login passwords on behalf of authorized users.

Figure 2.5   *SAS Federation Server Accounts*

The SAS Federation Server System User Account

About the System User Account

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.

A SAS Federation Server System User Account, **sasfedadm**, is created during installation of SAS Federation Server. This account is a member of the Federation Server Administrators group.

Activities Associated with the System User

The system user should identify users who will be administrators of SAS Federation Server, and grant them administrative privileges. There are two ways to grant users administrative privileges:

• Add the user to the SAS Federation Server Administrators group. This group has the administer privilege already assigned.

• Grant the user ADMINISTER privilege on the server object using administration DDL.

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 can have privileges granted or denied. 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.

The Administrator Account and Federation Server Administrators Group

About the Administrator Account

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

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

You can assign administrators by adding users to the Federation Server Administrators Group on SAS Metadata Server, or by granting the ADMINISTER privilege using the GRANT statement. However, only a system user can invoke the GRANT ADMINISTER DDL statement.

Adding a User to the Federation Server Administrators Group

With the addition of SAS Metadata Server in 4.2, you can grant users administrator privileges by adding them to the Federation Server Administrators group. Use the following procedure to designate a user as an administrator of SAS Federation Server.

1. Using SAS Management Console, navigate to the Federation Server Administrators group by selecting **Environment Management** ⇨ **User Manager** and select the **Federation Server Administrators** group in the right pane.

2. Open **Federation Server Administrators Properties** and select the **Members** tab.

3. Select a user from **Available Identities** and click the arrow to move the user object to **Current Members** of the Federation Server Administrators group.

Figure 2.6 *Federation Server Administration Properties*
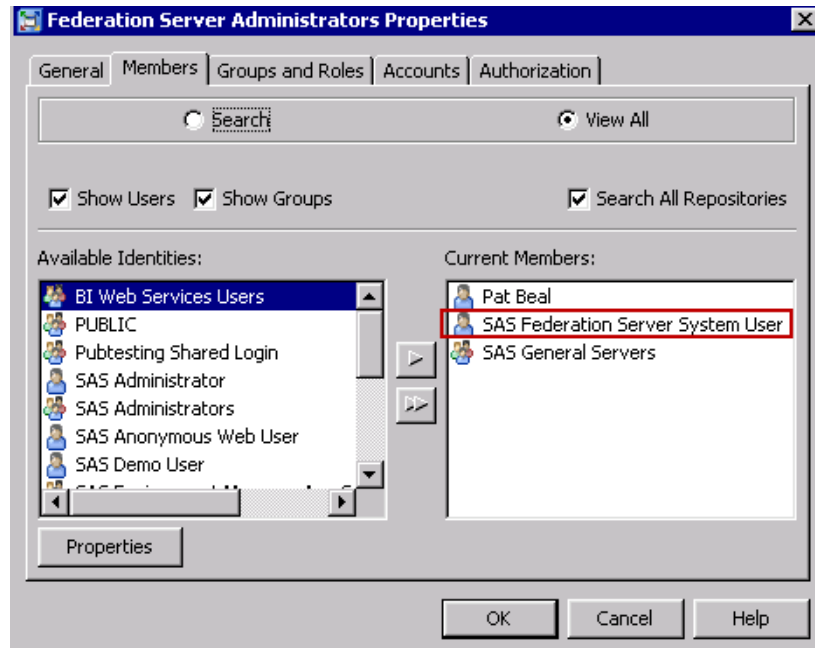
4. Click **OK** when you are finished adding users.

Setting the ADMINISTER Privilege Using DDL

Only system users can grant the ADMINISTER privilege using DDL. 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"
```

The example below grants the **ADMINISTER** privilege to the **user1** account on federation server, **FedServer1**:

```
GRANT administer ON FedServer1 TO "user1"
```

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

The SAS Trusted User Account

About the Trusted User Account

The **SAS Trusted User Account** account, **sastrust@saspw**, is created during installation of SAS Federation Server. A trust relationship is required for certain features, such as definer's rights views.

Here are a few key items 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.

- SAS Federation Server uses the trusted user account to connect to SAS Metadata Server in certain scenarios like definers rights views. This account is never used to log on to a server or application.

- The trusted user should not be a system user or administrator for SAS Metadata Server or SAS Federation Server.

Configure a License for SAS Federation Server

Overview

SAS licenses are SAS installation data (SID) files that are located in the **sid_files** directory of the SAS Software Depot or media. Copy the SID file(s) to a permanent location that can be accessed by the server such as the **/etc/license** directory. On UNIX, each SID file has a **UNX** suffix. To identify which license to apply, you must open the file and determine which products that SID file unlocks. Each server has its own unique SID file.

A license for SAS Federation Server is configured in the dfs_entities.dtd file. The license information is then propagated to the license option set in dfs_serv_common.xml. When you acquire a new license or the location changes for your existing license, you must update dfs_entities.dtd to reflect the new license information. See the Configuration Reference for details about dfs_entities.dtd and the license option set.

Configure a License on Windows

If the location of the license file has been moved after the initial installation, or if you acquired a new license file, follow these instructions to apply the license. The license file must reside in a directory that is accessible by the server, such as **etc/license**, located in the configuration path.

1. Open Windows Explorer and navigate to the **\etc** directory of the configuration path, which is commonly located at **SAS\Config\Levn\FederationServer**.

2. Open dfs_entities.dtd for editing and locate **<!ENTITY cfg.license.loc>** under **Common Configuration Parameters**. Here is an example of the entry:

   ```
   <!ENTITY cfg.license.loc      "$loc">
   ```

3. Update this entity with the location and name of your license file, as shown in the following example:

   ```
   <!ENTITY cfg.license.loc  "C:\temp\sid.txt">
   ```

Configure a License on UNIX

If the location of the license file has been moved after initial installation, or if you acquired a new license file, follow these instructions to apply the license. The license file must reside in a directory that is accessible by the server, such as **/etc/license**, located in the configuration path.

1. Navigate to the **/etc** directory of the configuration path, which is commonly located at **SAS/Config/Levn/FederationServer/**.

2. Open dfs_entities.dtd for editing and locate **<!ENTITY cfg.license.loc>** under **Common Configuration Parameters**. Here is an example of the entry:

   ```
   <!ENTITY cfg.license.loc      "$loc">
   ```

3. Update the **"$loc"** parameter with the location and name of your license file, as shown in the following example:

```
<!ENTITY cfg.license.loc  "/temp/sid.txt">
```

Configure Temporary Storage for SAS Utility Files

Overview

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.

Utility files are not compressed and can contain sensitive information. Therefore, restrict access to these files and store them in an appropriately protected subdirectory. Access to utility files should be limited to the process that created them.

The Directory Location

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_machine-name** 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 **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.
5. Click **OK** and start the Federation Server.

When SAS Federation Server is started, you should see a directory similar to **SAS_util000100000EF0_machine-name** 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, in Windows, **C:\Users\user_1\AppData\Local\Temp**.

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. Some of these items are configured using SAS Management Console.

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
`C:\Program Files \SASHome \SASFederationServer \4.2`	Installer, Administrator	Full Control
	Process user	Read and Execute, List Folder Contents
`C:\Program Files \SASHome \SASFederationServer \4.2`		
`C:\SAS\Config\Levn \FederationServer`		
`C:\Program Files \SASHome \SASFederationServer \4.2\var`	Installer, Administrator	Full Control
	Process user	Read, Write, List Folder Contents
`C:\SAS\Config\Levn \FederationServer\var`	The user who backs up SAS Federation Server; Backup Administrator	Read, List Folder Contents
TranPath as specified in the server configuration file `dfs_serv_common.xml`.	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.

Starting and Stopping the Windows Service

The SAS Federation Server runs as a Windows service that is accessible through the Control Panel or Management Console. To access the service:

1. Select **Start ⇨ Settings ⇨ Control Panel**

2. Double-click to open **Administrative Tools**, and select **Computer Management**.

3. Expand the **Services and Applications** folder.

4. Select **Services**, and **SAS Federation Server**.

5. Select either **Stop the service** or **Restart the service**.

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

`/installation_root/`
`SASHome`
 	
`/SAS/Config/Levn/`
`FederationServer`
	Installer, Administrator	Read, Write, Execute

	Process user	Read, Execute

`/installation_root/`
`SASHome/`
`FederationServer/var`
 	
`/SAS/Config/Levn/`
`FederationServer/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 drivers, you must set environment variables as outlined in the following sections.

Set the LANG Environment Variable

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

Most UNIX or Linux systems use the **LANG** environment 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 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 Data Sources

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

• The version or release of the client shared libraries installed on your operating system. This is important due to potential incompatibilities between DBMS versions or releases.

• The location of the 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.

SAS Federation Server Driver for Apache Hive

Hadoop JAR files must be installed and the SAS_HADOOP_JAR_PATH environment variable defined before using the Driver for Hive. This environment variable is set during installation if Hadoop is included in the plan. The variable points to the location of the Hadoop JAR files, and is defined using the **SetEnv** option set in the **dfs_serv.xml** configuration file. Here is an example:

```
<OptionSet name="SetEnv">
      <Option name="SAS_HADOOP_JAR_PATH">\SAS\Config\Lev1\FederationServer
       \lib\Hadoop</Option>
 </OptionSet>
```

If the JAR file location changes, you must update the SAS Federation Server configuration file with the new location.

SAS Federation Server Driver for DB2

The SAS Federation Server Driver for DB2 uses shared libraries that are referenced as shared objects in UNIX. 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 Greenplum

To use ODBC with Greenplum, you must set the **ODBCINI** environment variable to the **odbc.ini** file located in the Federation Server installation path:

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

When you run **dfsadmin**, the **ODBCINST** environment variable is set to the **odbcinst.ini** file located in the federation server installation path. Here is an example:

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

SAS Federation Server Driver for Netezza

The Netezza ODBC drivers are ODBC API-compliant shared libraries that are referenced as shared objects in UNIX. You must include the full path to the shared libraries in the shared library path as shown below so that the ODBC drivers can be loaded dynamically at run time.

	AIX
Bourne Shell	`$ LIBPATH=$ODBCHOME/lib64:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH $ODBCHOME/lib64:$(LIBPATH)`

	HP-UX for the Itanium Processor Family Architecture
Bourne Shell	`$ SHLIB_PATH=$ODBCHOME/lib64:$SHLIB_PATH` `$ export SHLIB_PATH`
C Shell	`$ setenv SHLIB_PATH $ODBCHOME/lib64:$(SHLIB_PATH)`

	Linux for Intel Architecture, Linux for x64, Solaris, and Solaris for x64
Bourne Shell	`$LD_LIBRARY_PATH=$ODBCHOME/lib64:$LD_LIBRARY_PATH` `$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH $ODBCHOME/lib64:$(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 for specific configuration information.

The Drivers for ODBC are ODBC API–compliant shared libraries, referred to as shared
objects in UNIX. 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}`

SAS Federation Server Driver for Oracle

You can connect to any Oracle server from SAS Federation Server (Driver for Oracle) using the SAS Federation Server Driver for Oracle. Refer to SAS System Requirements for the supported releases of the Oracle client.

To use the 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 the location of the Oracle shared libraries. This is required since the driver 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 SAP

SAP® software requires extensive configuration before it can be used. For more information, see "Installing and Configuring the SAS Federation Server Driver for SAP" .

SAS Federation Server Driver for SAP HANA

The SAS Federation Server Driver for SAP HANA uses an ODBC interface to access SAP HANA. The SAS Federation Server Driver for SAP HANA requires the 64–bit ODBC driver for SAP HANA. The SAP HANA client includes the ODBC driver.

These are the prerequisites for configuration of the SAS Federation Server Driver for SAP HANA:

- You have downloaded the SAP HANA client software from SAP Service Marketplace and installed and configured the ODBC driver.

- For more information about how to obtain the software, see the SAP HANA Master Guide on http://help.sap.com/hana_appliance/.

- For information how to install and configure the ODBC driver refer to the SAP HANA Client Installation Guide on http://help.sap.com/hana_appliance/.

- You must include the full path to the shared library in the shared library path so that the ODBC drivers can load dynamically at run time.

AIX	

Bourne Shell	`$ LIBPATH=/usr/sap/hdbclient:$LIBPATH`
	
	`$ export LIBPATH`

C Shell	`$ setenv LIBPATH /usr/sap/hdbclient:$LIBPATH`

HP-UX for the Itanium Processor Family	

Bourne Shell	`$ SHLIB_PATH=/usr/sap/hdbclient:$SHLIB_PATH`
	
	`$ export SHLIB_PATH`

C Shell	`$ setenv SHLIB_PATH /usr/sap/hdbclient:$SHLIB_PATH`

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

Bourne Shell	`$ LD_LIBRARY_PATH=/usr/sap/hdbclient:$LD_LIBRARY_PATH`
	`$ export LD_LIBRARY_PATH`
C Shell	`$ setenv LD_LIBRARY_PATH /usr/sap/hdbclient:$LD_LIBRARY_PATH`

The SAS Federation Server Driver for SAP HANA can use data sources defined in the odbc.ini file to identify the SAP HANA server. The general format of the odbc.ini is:

```
[ODBC Data Source]
SERVERNODE=hana_host:hana_port
```

For example:

```
[SAPHHANADSN]
SERVERNODE=hanasrv1.mycompany.com:30015
```

Set the ODBCINI environment variable to the location and name of your odbc.ini:

Bourne Shell	**ODBCINI=path-to/odbc.iniexport ODBCINI**
C Shell	**setenv ODBCINI path-to/odbc.ini**

SAS Federation Server Driver for Teradata
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`

Configuring ODBC Connections

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 using 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 before attempting connection to 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.

Note: The unixODBC driver manager cannot be used alongside the UNIX driver manager for wire protocol drivers that are shipped with SAS Federation Server.

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 ODBCINI environment variable accordingly.

3. **DM_UNICODE=utf-16** is required in the advanced options of the ODBC data service that is used with the driver manager.

Note: The **DM_UNICODE** connection option is also available with the "SAS Federation Server Driver for ODBC".

Use the vendor supplied client configuration utility for non-ODBC connections. For more information about configuring third-party databases, See "Setting Environment Variables for Data Sources" .

SAS Federation Server Configuration Reference

Locale Support

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

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

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

Key Configuration Files

These configuration files are located in the configuration directory of SAS Federation Server, for example, **C:\SAS\Config\Lev1\FederationServer\etc**. The following table lists the key configuration files for SAS Federation Server:

Type	File or Script Name	Description
Data Quality	**dfs_serv_dq.xml**	This is the configuration file that contains the data quality methods and the location of SAS QKB.
Server	**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. Detailed configuration information is presented in the Configuration Options on page 37.
Server DTD	**dfs_entities.dtd**	The **dfs_entities.dtd** file contains the values that were supplied during installation of SAS Federation Server. These values are referenced by other configuration files, as **dfs_serv.xml** and **dfs_serv_common.xml** files.

Type	File or Script Name	Description
Logging	**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 located in the **/etc** directory of the configuration path.
	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 configuration path.

About dfs_entities.dtd

SAS Federation Server uses the dfs_entities.dtd file to store values that are supplied during installation and configuration. 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. Following is an example of the dfs_entities file. Note that SAS Federation Server now uses a separate path/directory for installation and configuration:

Figure 3.1 SAS Federation Server Configuration Parameters – dfs_entities.dtd

```
<!-- Common configuration parameters -->
<!ENTITY cfg.home.config            "C:\SAS\Config\Lev1\FederationServer">
<!ENTITY cfg.home.install           "C:\Program Files\SASHome
\SASFederationServer\4.2">
<!ENTITY cfg.license.loc            "C:\temp\sid.txt">
<!ENTITY cfg.fsid                   "Federation Server-pubtesting-Logical">
<!ENTITY cfg.logs.dir               "C:\SAS\Config\Lev1\FederationServer
\var\Logs">
<!ENTITY cfg.fsadmin                "Federation Server Administrators">
<!ENTITY cfg.SASHOME                "C:\Program Files\SASHome">
<!ENTITY cfg.installdir             "C:\Program Files\SASHome
\SASFederationServer\4.2\bin">
<!ENTITY cfg.tkjava                  "C:\Program Files\SASHome
\SASFoundation\9.4\tkjava\sasmisc">
```

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, also referred to as 'options', specify a **name=value** pair as configuration file options. Option names can stand alone in a configuration file, or they are contained within an option set. Here are the different types of option name configurations that appear in SAS Federation Server configuration files:

The string that follows is a simple **name=value** pair that represents a configuration option:

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

In the example that follows, port is the option name that you are configuring and 21030 is the specified port number for the server:

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

Option Sets

An option set is a collection of one or more options, or option names. Options that belong in an option set will not be assessed correctly if they are not placed within the opening and closing tags of the <OptionSet>. An option set requires that you specify at least one option for the configuration to be valid. Here is an example:

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

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 exists for 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.

CAUTION:
Server configurations that are set in the system configuration files will override existing configurations on SAS Metadata Server.

SAS Metadata / SAS Federation Server Configuration Options

This table specifies the required configuration options for connectivity from SAS Federation Server to SAS Metadata Server.

Name	Description	Configuration File
MetaConfig	**MetaConfig** specifies the path to the **sasv9_meta.cfg** file that is configured and copied to SAS Federation Server by the SAS Deployment Wizard process. The **sasv9_meta.cfg** file contains the metadata user name and password information needed to complete the connection to SAS Metadata Server. `<Option name="MetaConfig">path-to-sasv9_meta.cfg-file</Option>`	dfs_serv.xml
MetaProfile	MetaProfile is used to connect to SAS Metadata Server using a profile. MetaProfile specifies the path to the metadataconfig.xml file which contains connection information to SAS Metadata Server. MetaUser and MetaPass specifies the name and password that you are connecting with. `<Option name="MetaProfile">path to metadataConfig.xml</Option>`	dfs_serv.xml

MetaUser
MetaPass
	MetaUser and MetaPass specifies the user name and password used to connect to SAS Federation Server.
	
	`<Option name="MetaUser">userid</Option>`
	`<Option name="MetaPass">password<Option>`
		dfs_serv.xml

SAS Federation Server Configuration Options

This table specifies the configuration options for the SAS Federation Server environment.

Name	Description	Configuration File

Authentication Provider Domain	The AuthProviderDomain option associates authentication providers with domains. This option reroutes users from their default provider and domain to the **tksecas** provider for validation. The **tksecas** provider then forwards the authenticating user to SAS Metadata Server for resolution and provides SAS Federation Server with the identity of the connected user. The **tksecas** provider accepts multiple domains. The syntax for this option is a series of comma-separated mappings: **provider-name:domain-name.**
	
	`<Option name="AuthProviderDomain">( SASPassword:, tksecas:saspw, SASGenerated:,`
	`tksecas:'!*(generatedpassworddomain)*!' )</Option>`
		dfs_serv.xml

Application Name	Application name specifies a name for the SAS Federation Server. The default is set in dfs_entities.dtd as the name of the logical federation server definition in SAS Metadata Server. This option corresponds to the **X{App.Name}** entry in the SQL logging configuration file. `<Option type="String" name="env:App.Name">&cfg.fsid;</Option>`	dfs_serv.xml from dfs_entities.dtd

Set Environment Variables	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 to use in the event that the default database directory runs out of space. Once the default directory has no available space, the engine switches to the directory specified in FIREBIRD_TMP.
	
	`<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>`
	
	Note: The SetEnv option set is also used to define the path for Hadoop JAR files. See "Hadoop Configuration Options" below.
		dfs_serv_common.xml

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

Name	Description	Configuration File

Security Provider	The security provider option set provides information about SAS Federation Server's security provider, including the threaded kernel extension name and other information specific to the security provider.
	
	• Database: Specifies the name of the transactional data store. The default name is SYSCAT.
	
	• ServerComponent: Specifies the name of the Metadata Server object that identifies the data management server, machine, and port of the server where the system catalog (SYSCAT) resides.
	
	`<OptionSet name="SecurityProvider">`
	`<Option name="extension">tkescfb</Option>`
	`<Option name="Database">syscat</Option>`
	`</OptionSet>`
		dfs_serv_common.xml

Function Dispatch Manager Option	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**. `<Option name="FunctionDispatchManager">tktsfd</Option>`	dfs_serv_common.xml
Content Root Option	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. 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 do not match are rejected. • 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. `<Option name="ContentRoot">content_root_path</Option>`	dfs_serv_common.xml

License Option Set	Sets the location of the license for SAS Federation Server.
	
	`<OptionSet name="License">`
	`    <OptionSet name="Primary">`
	`        <Option name="Location">&cfg.license.primary.loc</Option>`
	`    </OptionSet>`
		dfs_serv_common.xml from dfs_entities.dtd

Name	Description	Configuration File
Transactional Data Store Options	The FIREBIRD environment variable specifies the location of the Transactional Data Store installation files.	dfs_serv_common.xml
	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.	
	`<Option name="FIREBIRD">drive:\install_dir\lib\fbembed</Option>` `<Option name="FIREBIRD_LOG">drive:\install_dir\var\log</Option>`	

Hadoop Configuration Options

Name	Description	Configuration File
Path to Hadoop JAR Files	As a prerequisite for the using the SAS Federation Driver for Apache Hive, Hadoop JAR files must be installed and the SAS_HADOOP_JAR_PATH environment variable defined before using the driver. The variable points to the location of the Hadoop JAR files and is defined in the SetEnv option set during installation of SAS Federation Server, if Hadoop is included with the order.	dfs_serv.xml from dfs_entities.dtd

	`<OptionSet name="SetEnv">`
	`    <Option name="SAS_HADOOP_JAR_PATH">\SAS\Config\Lev1\FederationServer`
	`    \lib\Hadoop</Option>`
	` </OptionSet>`	

Hadoop Configuration Path	Hadoop cluster configuration files include core-site.xml, hdfs-site.xml, hive-site.xml, mapred-site.xml, and, if applicable, yarn-site.xml. You must copy Hadoop configuration files from the Hadoop cluster to a physical location accessible by SAS Federation Server, if they are not already accessible. You must also define and set the environment variable SAS_HADOOP_CONFIG_PATH to the location of the Hadoop configuration files. By defining the environment variable as a configuration option, you do not need to specify HD_CONFIG in the connection string.	dfs_serv.xml
	`<Option name="SAS_HADOOP_CONFIG_PATH">\\host-name\path\hadoop-config\</Option>`	

Data Quality Functions Configuration

Name	Description	Configuration File
Data quality functions configuration	Specifies the location of the Quality Knowledge Base (QKB) that is used with the data quality functions.	dfs_serv_dq.xml from dfs_entities.dtd
	`<!ENTITY cfg.qkb.loc "\\path to QKB directory\QKB\CI24"`	

Miscellaneous Configuration Options

Name	Description

Append Environment Variable OptionSet	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. If a delimiter is needed, it should be included at the beginning of the specified value.
	
	 `<OptionSet name="AppendEnv">`
	`   <Option name="FIREBIRD">`*drive*`:\install_loc\firebird</Option>`
	`</OptionSet>`

Deadlock Protection Option	Specifies the wait time for a connection in deadlock. A deadlock is sometimes caused by competing resources on the server resulting in perpetual wait time for the tasks to complete. This option controls the wait time in milliseconds before timing out a lock attempt that is causing the deadlock. If a connection cannot be acquired within the specified time limit, the request fails and the deadlock connection is released, allowing the remaining connection to run to completion. The default is **<=0** which waits 'forever'. `<Option name="SystemDBCTimeOut">milliseconds</Option>`
Memory Size Option	The MemSize 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. `<Option name="MemSize">nnnnn [(K|k|M|m|T|t)][(B|b)]]</Option>`

System Users Option	The System Users option defines the system user account that is 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. A system user should be a domain-qualified user name.
	
	 `<SystemUsers>`
	`   <Option name="Account">domain\uid1</Option>`
	`   <Option name="Account">domain\uid2</Option>`
	`</SystemUsers>`
	
	 Note: A system user account, sasfedadm, is created during SAS Federation Server installation

Select * Expansion	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. `<Option name="SelectStarExpansion">ALL</Option>`

Chapter 4

SAS Federation Server Administration

Overview

User Account Administration

User account administration is performed using SAS Management Console. SAS Management Console is a Java application that provides a single point of control for administering your SAS servers and for managing metadata objects that are used throughout with SAS Federation Server. Whenever the SAS Metadata Server is running, you can use SAS Management Console to connect to the SAS Metadata Server and view and manage the objects that are stored in the server's metadata repositories. See *SAS Management Console: Guide to Users and Permissions* for additional information.

Server Administration and Backups

SAS Metadata Server backups are facilitated using the Deployment Backup and Recovery Tool. The Deployment Backup and Recovery tool provides an integrated method for backing up and recovering your SAS content across multiple tiers and machines. The tool is installed on the middle tier as part of the SAS Web Infrastructure Platform.

By default, metadata server backups are scheduled to run at 1:00 a.m. server local time every day except Sunday. The Deployment Backup and Recovery tool, if it is configured, backs up the metadata server (along with other resources) each Sunday at 1:00 a.m. by default. Administrators can use SAS Management Console to change the metadata server backup schedule and configuration options, including the backup directory location and the backup retention policy. Backups can also be run on an unscheduled basis from SAS Management Console, from the operating system command line, from SAS, or through third-party scheduling software. For additional information, refer to *SAS Intelligence Platform: System Administration Guide*, "Using the Deployment Backup and Recovery Tool".

The Deployment Backup and Recovery Tool does not backup SAS Federation Server's system catalog (SYSCAT) and associated content. However, you can perform backup of SAS Federation Server with the backup utility, dfsutil. See "Utilities for SAS Federation Server " for additional information.

Data Management and Administration

SAS Federation Server data management, including data access privileges, is administered with the use of various DDL statements and SQL commands. You can also accomplish data management tasks with 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. Most of the functions performed with administration DDL, can be accomplished using SAS Federation Server Manager.

Utilities for SAS Federation Server

Introduction

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

UNIX Utilities

Overview

UNIX server utilities are located in the **/bin** directory of the server's configuration path.

Utility Name	Function
dfsadmin	**./bin/dfsadmin start│stop│status│restart**
	Use for server administration.
dfsutil	**./bin/dfsutil backup│restore**
	Use for database backup and restore. The options are **backup** and **restore**.
dfdbconf	**./bin/dfdbconf A│D│X**
	ODBC configuration tool. Use to add new data sources or edit existing ones. The options are A –Add, D –Delete, and X – Exit.
dfdbview	ODBC viewer used to list data sources: **./bin/dfdbview –l(ist)**
	To test data sources and run SQL: **./bin/dfdbview DSN-name**

dfsadmin – Server Administration

SAS Federation Server for UNIX contains the **dfsadmin** utility, located in the **/bin** directory of the server's configuration path. Run any of the commands using the following syntax: **./bin/dfsadminyourcommand** where *yourcommand* is one of the following:

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.

restart
　　Restarts SAS Federation Server.

dfsutil – Database Backup and Restore

It is recommended that you back up the SAS Federation Server databases periodically, especially the system catalog, SYSCAT.tdb. The **dfsutil** utility is located in the **/bin** directory of the server's configuration path. With **dfsutil**, you can perform dynamic database backups without disruption to server operations. You can back up the system database (SYSCAT.tdb) and the default SQL logging database (SQL_Log). However, you cannot back up the SQL logging database with **dfsutil** if it uses a third-party data store. You can also restore databases using **dfsutil** if the database was backed up using the same utility.

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

The following procedures describe how to use **dfsutil** to perform a backup and restore for a SAS Federation Server database. Note that you can run backups while the server is running.

Backup

Back up a SAS Federation Server database using the dfsutil command: UNIX

To back up a database, use the **dfsutil** command with the **-db** parameter. Note that you can run this command while the server is operational. Using the configuration path, navigate to the directory that contains **/bin** and use the following command syntax:

```
.bin/dfsutil backup -db syscat /path_to_backup
```

Restore

Restore a SAS Federation Server database using the dfsutil command: UNIX

To restore a database, use the **dfsutil** command with the **-db** parameter. Using the configuration path, navigate to the directory that contains **/bin** and use the following command syntax:

```
.bin/dfsutil restore -db syscat /path_to_backup
```

Note: SAS Federation Server must be shut down prior to running a database restore.

dfdbconf – ODBC Configuration

Use **dfdbconf** to add a new data source or edit existing data sources in your ODBC configuration.

To add a new data source:

1. Run the following command from the SAS installation directory: **./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.

dfdbview – List and Test Data Sources

Use **dfdbview** to list data sources and run interactive SQL queries.

• To list your configured data sources, run the following command: **dfdbview -l**.

- Use the following command to connect to a data source and run SQL: **./bin/dfdbview DSN-name**. For example, if you added a data source called my_oracle, run **./bin/dfdbview my_oracle**. You might be prompted for a user name and password if there is additional security on the DSN. After establishing connection to the data source, 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.

Windows Utilities

dfsutil – Database Backup and Restore

It is recommended that you back up the Federation Server databases periodically, especially the system catalog, SYSCAT.tdb. Use **dfsutil** to back up and restore the system catalog. This utility is located in **\bin** of the configuration directory (for example, **<drive>\SAS\config\Levn\FederationServer\bin**). With **dfsutil**, you can perform dynamic database backups without disruption to server operations. If you back up a database with **dfsutil**, then you are required to restore that database using **dfsutil**.

Backup

Back up a SAS Federation Server database using the dfsutil command: Windows

To back up a database, use the **-db** parameter with the **dfsutil** command and include the name of the database in the backup statement. You do not need to stop the SAS Federation Server service to run this command. Here is the command syntax: **drive:\SAS\config\Levn\FederationServer\bin>dfsutil backup -db syscat path_to_backup_directory\backup_filename**

Here is an example backup command: **c:\SAS\config\Lev1\FederationServer\bin>dfsutil backup -db syscat c:\dfsutilBackup\syscatbk.tdb**

Restore

Restore a SAS Federation Server database using the dfsutil command: Windows

You can restore a database using **dfsutil** only if the database was backed up with dfsutil. To restore a database, use the **-db** parameter with the **dfsutil** command. Here is the command syntax: **drive:\SAS\config\Levn\FederationServer\bin>dfsutil restore -db syscat\path_to_backup_directory\syscat.tdb**

Here is an example restore command:

C:\SAS\config\Lev1\FederationServer\bin>dfsutil restore -db syscat\c:\dfsutilBackup\syscatbk.tdb

Note: All services for SAS Federation Server must be shut down prior to running a database restore.

SQL Scripting for SAS Federation Server Administration

Overview

SAS Federation Server provides SQL language scripting capabilities to handle administrative needs for start-up and shutdown events. Administrators can write and execute scripts to manage auditing or related event notifications. SQL scripts execute in one of two phases: **Startup.Epilog** and **Stop.Prolog**. To run a script, add the name of the script to the server's configuration file, **dfs_serv.xml**.

Additional configuration information, including an example, appears at the end of this topic.

About the Configuration for SQL Scripting

XML Format

SQL scripts are specified in an XML configuration file as an SQL node in an **OptionSet** element that includes a name attribute of SQL:

```
<OptionSet name="SQL">
...
</OptionSet>
```

An option set can consist of one or more option names. Option names that belong in an OptionSet will not be assessed correctly if they are placed outside of the OptionSet.

Scripts are arranged in a hierarchical, parent-child format. Here is an example of a parent option set containing two child options. The child option sets are siblings to each other:

```
<OptionSet name="SQL">
        ...
        <Option name="Command">command 1</Option>
        <Option name="Command">command 2</Option>
        ...
    </OptionSet>
```

SQL nodes can be nested with the outer-most node, which is the script. Nested nodes correspond to SQL commands that run within the script. Additional XML elements provide execution context for the SQL nodes. Here is an example of nested nodes:

```
<OptionSet name="SQL">
        <Option name="Condition">SQL Boolean scalar result query</Option>
    <OptionSet name="SQL">
        <Option name="Command">command 1</Option>
        <Option name="Command">command 2</Option>
        ...
    </OptionSet>
    ...
    <OptionSet name="SQL">
        ...
        <Option name="Command">command 1</Option>
        <Option name="Command">command 2</Option>
```

```
            ...
        </OptionSet>
</OptionSet>
```

The nested sibling SQL nodes are highlighted in gray. The commands of each of the inner SQL nodes are run according to the specified error mediation if the condition is true in the outer (parent) SQL node.

Elements of SQL Scripting

Listed below are the valid elements within an SQL node.

SQL Node

`<OptionSet name="SQL">...</OptionSet>`

The OptionSet specifies an SQL node containing a nested SQL script composed of other elements, each specified within an **`OptionName`**.

Name

`<Option name="name">SQL node name or description</Option>`

Text that specifies the name of the SQL node. This option is used for logging context.

SQL Error Mediation Action

`<Option name="SQLErrorMediationAction">Error mediation action</Option>`

Specifies the required action that is needed to mediate errors during SQL command execution:

STOP
Specifies that the script terminate execution when it encounters an error that satisfies the current SQL state and match mode criteria. **STOP** is the default mediation action.

CONTINUE
Specifies that the script continue executing the next SQL node regardless of encountering an error that satisfies the current SQL state and match mode criteria. This is used when a set of SQL commands should run without regard to the success of those commands previously executed within the same SQL node.

Error mediation for the SQL node is inherited from the nesting SQL node, if it exists.

SQL Error Mediation SQL State Match Mode

`<Option name="SQLErrorMediationSSMatchMode">SS match mode</Option>`

Specifies the SQL state match mode used to identify specific SQL states requiring mediation action:

EXCLUDE
Action taken when the SQL state does not match one of the states specified within the set of scoped states of the current SQL node.

INCLUDE
Action taken when the SQL state matches one of the states specified within the set of scoped states of the current SQL node.

IGNORE

Specifies that SQL states within scope are ignored. This causes the mediation action to be honored based on the success or failure of the command regardless of the SQL state. IGNORE is the default match mode.

The SQL state match mode of the SQL node, which is inherited from the nesting SQL node, if it exists.

Phase

`<Option name="Phase">Server phase</Option>`

SQL scripts execute in one of two phases: `Startup.Epilog` and `Stop.Prolog`.

Startup.Epilog

`Startup.Epilog` is executed on start-up of SAS Federation Server before listening for connections and after administration DDL can be executed. These scripts can connect to any configured data service and execute SQL that is run as the process user.

Stop.Prolog

`Stop.Prolog` is executed on SAS Federation Server shutdown after quiescing or dropping client connections. These scripts can connect to any configured data service and execute SQL that is run as the process user.

SQL State

`<Option name="SQLState">SQL state expression</Option>`

Specifies an SQL state or prefix that is used to identify specific SQL states requiring error mediation. An SQL state can be specified as a full 5–character mnemonic such as `HY001`, or as a prefix matching any SQL state starting with a specified prefix. In addition, a leading **+** character (concatenation operator) can be prepended to the mnemonic to add the SQL state to the current set of constraining SQL states. Without the leading concatenation operator, the specified SQL states replaces any SQL states that were previously specified.

The SQL states for the SQL node are inherited from the nesting SQL node if any.

Connection String

`<Option name="ConnectionString">connection-string</Option>`

Specifies the connection string that is used to access the data source to which SQL commands are submitted for execution.

Command

`<Option name="Command">SQL command</Option>`

Specifies the SQL command to execute. The status and SQL state of the execution is processed according to the current error mitigation configured in the containing SQL node lineage.

Nested parameterized SQL commands can be specified where the inner command is executed once for each row materialized in the outer command's result set. Parameters are specified using **@n** syntax where **n** is a single parameter number corresponding to the n'th column of the outer command's result set. Note that commands can be nested only once. Here is an example of the nested command:

```
<OptionSet name="SQL">
...
<Option name="Command">outer query command</OptionSet>
<OptionSet name="SQL">
...
<Option name="Command">parameterized inner command 1</Option>
</OptionSet>
...
<OptionSet name="SQL">
...
<Option name="Command">parameterized inner command n</Option>
</OptionSet>
</OptionSet>
```

Condition

`<Option name="Condition">SQL Boolean scalar result query</Option>`

Specifies a query that resolves to a scalar Boolean result. Nonconforming commands will fail, causing script execution to end. If the command result is **1**, all sibling SQL nodes are executed. Otherwise, they are skipped. Only the first condition is processed within an SQL node parent.

As a precursor to executing a set of SQL commands, a condition can be used to check for the existence of a table, a row within a table, or a value within a row. A condition on the outermost level SQL node will effectively make the entire script's execution dependent on the result of the specified query.

The following example condition returns **1** when table T has at least one row matching the **WHERE** clause, which is not shown. If the query returns **1**, sibling SQL nodes contained in the parent SQL node are executed:

```
<Option name="Condition"> select cast(case when count(*) > 0 then 1
  else 0 end as integer) from T where ...</Option>
```

Auto Commit

`<Option name="AutoCommit">value</Option>`

Use autocommit to create a block of SQL that executes under a single transaction. The options are **true** (default) and **false**.

TRUE

When autocommit is set as true, each statement is executed as its own transaction and there is no rollback.

FALSE

When autocommit is set as false, the transaction is committed or rolled back depending on the SQL state at the end of the block of SQL statements. If no errors have occurred, or errors are permitted by the settings of the SQL script, the transaction is committed. Otherwise, it is rolled back.

Example SQL Script

This example script copies the content of in-memory MDS tables to a persistent data store when the server is stopped. You can use an inverse script to load the tables back

into the MDS service in the **`Startup.Epilog`** phase. Error mediation prevents
CREATE TABLE commands from stopping script execution when the table already
exists in the data store. Errors with SQL states beginning with **42S** are excluded from
the stop action. Also, if the C_STORE_MDS catalog requires credentials to connect, you
can supply those in the connection string.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<OptionSet name="SQL">
   <!--
   Default phase, SQL error mediation control
   -->
   <Option name="name">MDS to STORE_MDS store</Option>
   <Option name="SQLErrorMediationAction">STOP</Option>
   <Option name="SQLErrorMediationSSMatchMode">EXCLUDE</Option>


   <!--
   MDS Store script
   -->
   <OptionSet name="SQL">
    <Option name="Phase">Stop.Prolog</Option>
    <Option name="SQLState">42S</Option>
    <Option name="ConnectionString">
    driver=FEDSQL;conopts=((security=NO;catalog=C_STORE_MDS);
                           (security=NO;catalog=C_MDS))
   </Option>
   <!--
   Result set generator command: Enumerate all MDS tables...
   -->
   <Option name="Command">
   select TABLE_SCHEM,
          TABLE_NAME
   from DICTIONARY.TABLES
      where TABLE_CAT='C_MDS' and
            TABLE_TYPE='TABLE'
   </Option>
   <!--
   Result set iterator command:
   Store MDS catalog tables in C_STORE_MDS "mirror" catalog
   -->
   <OptionSet name="SQL">
    <Option name="Command">
    create table C_STORE_MDS.S."@2" as
       select * from C_MDS."@1"."@2" where 1=0
    </Option>
   </OptionSet>
   <OptionSet name="SQL">
    <Option name="Command">
    delete from C_STORE_MDS.S."@2"
    </Option>
   </OptionSet>

   <OptionSet name="SQL">
    <Option name="Command">
    insert into C_STORE_MDS.S."@2"
       select * from C_MDS."@1"."@2"
    </Option>
```

```
    </OptionSet>
  </OptionSet>
</OptionSet>
```

This example script was saved as **store_mds.xml**. After saving the script, edit the **dfs_serv.xml** configuration file as highlighted in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>


<DOCTYPE Config [
<ENTITY % entities SYSTEM "dfs_entities.dtd">
%entities;
<ENTITY MDS_SCRIPT SYSTEM "store_mds.xml">
]>


<Config name="TSConfig">

<!-- Common server options -->
&SERVER_COMMON;

<!-- Run the MDS script -->
&MDS_SCRIPT;



</Config>
```

For additional information about the **dfs_serv.xml** configuration file, see "SAS Federation Server Configuration Reference".

SAS Federation Server Database

Overview

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

The system catalog contains information about the configuration of SAS Federation Server. This information is returned to the user through queries against the database using information views.

Working with the SAS Federation Server Database

Creation of the System Tables

The SAS Federation Server database, also referred to as the system catalog, is created when SAS Federation Server is initially invoked. Upon invocation, 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, the system tables are 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 SYSCAT.tdb is **/install/cfgsas1/config/Lev1/ FederationServer/var**. The name and location SYSCAT.tdb are contained in the dfs_entities.dtd configuration file.

Changing the Database Location
The following configurations require updates in the event that the location of SYSCAT.tdb changes:

dfs_entities.dtd
<!ENTITY cfg.TRANPATH "c:\temp">
> Change the **cfg.TRANPATH** entity to point to the new location of SYSCAT.tdb. This configuration change updates dfs_serv_common.xml.

<!ENTITY cfg.FIREBIRD_LOCK "&cfg.TRANPATH;">
> Change the value for **cfg.FIREBIRD_LOCK** entity to point to the new location of SYSCAT.tdb. This configuration change updates the location of the database lock files for both the SQL_LOG and SYSCAT transactional databases in dfs_log_SQL_Logging.xml. When updating the FIREBIRD_LOCK environment variable, use an absolute path only.

dfs_log.xml
<!ENTITY DFS_DBAPPENDER_DB "&cfg.TRANPATH;/&cfg.sqllog;">]>
> Change the value of the **DFS_DBAPPENDER_DB** entity to point to the new location. This update changes the location of the SQL_LOG transactional database.

Database Backup and Restore

About dfsutil
It is recommended that you back up the Federation Server databases periodically, especially the system catalog, SYSCAT.tdb. Use **dfsutil** to back up and restore the system catalog. See "Utilities for SAS Federation Server " for additional information about dfsutil, and for backup and restore procedures.

High Availability

High availability is a failure response mechanism is to eliminate single points of failure in SAS Federation Server. A single point of failure is any component that would cause a service interruption should it become unavailable. The goal of high availability is to ensure redundancy for the components that are required for the proper function of SAS Federation Server which includes the system catalog.

While the database utilities do not allow for high availability of the SAS Federation Server database, there are methods that can serve as a way to maintain uptime of SAS Federation Server.

Database Recovery
> If the system fails as a result of database corruption, you can restore the database from the last backup. However, this scenario can result in lost data as a result of the corruption. The amount of lost data depends on the extent of the corruption. Restoring from backup can also be a time-consuming process.

Log Shipping
> Send a backup copy of the db to a readily-available server.

Open Source Solutions
use HAProxy.

SAS Federation Server Resource Cache

Overview

Authorization data that is used frequently can be cached from SAS Metadata Server and retained on SAS Federation Server until the cached information is refreshed or purged. This data cache can help improve server performance by reducing the number of calls needed from SAS Federation Server to SAS Metadata 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 the authenticating server. Several of these configurable caches are periodically repopulated as SAS Federation Server captures information from SAS Metadata Server during the authentication process. The cache names prefixed with AS represent an Authentication Service cache. By default, resources related to SAS Metadata 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" .

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

Cache Name	Description
AS	All SAS Metadata Serverauthentication service (AS) cached resources
AS.Name	Name to identifier mappings
AS.Name.Subjects	User name to SAS Metadata Server identifier cache
AS.Name.Groups	Group name to SAS Metadata Server identifier cache

Cache Name	Description
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 SQL Console to select from the Information Views;

```
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

All caches support the TIMEOUT option. The value for TIMEOUT specifies the length of time, in seconds, that a resource can be cached before being considered stale and marked for on-demand refresh. When a resource becomes stale, it is typically refreshed and reached on its next access. 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. 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 time-out of ResultSet to 300 seconds explicitly and also sets all children (for example, **ResultSet.View** to 300 seconds. Note that the statement only persists 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 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 named cache should be refreshed. Associated resources are reacquired and cached on next access and can be flushed immediately. This option is not persisted and using it does not affect existing properties that have already been configured for the named cache. 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 secure objects. This is the default privilege caching level.

CONTAINER   Cache privileges for schemas and all higher level secure objects.

NONE / OFF   Used to turn off all privilege caching.

Managing Client Connections

Connection Pooling

About Connection Pooling

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.

Configuring Connection Pooling

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

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.

To disable connection pooling, use **DROP_CONNECTION_POOLING**. Configuring **ENABLE_CONNECTION_POOLING** using a value of 0 (zero) is invalid and does not disable connection pooling.

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. To disable the time-out, use **DROP_CONNECTION_POOL_TIMEOUT**. Configuring **CONNECTION_POOL_TIMEOUT** using a value of 0 (zero) does not disable the time-out.

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 the 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.

Handling Client Disconnects

If a client should disconnect unexpectedly, for example, a client process is suddenly dropped, any work in progress will run to completion. Work in this context is a single SAS Federation Server API call, such as Execute or Fetch. Using SAS Federation Server Manager, identify the user's orphaned object and stop the session by performing the following tasks:

1. Select a SAS Federation Server object in the tree.

2. Select the **Current Connections** tab.

3. Use the drop-down list and select **Show Sessions**.

4. Select the session ID associated with the disconnected user and click **Close Session**.

SQL Logging

Overview

SQL Logging is the ability to view SQL statements submitted to SAS Federation Server. SQL statements can be combined with other information (for example, the user ID of the user who submitted the SQL, and information about prepare, execute, and cursor phases). Metrics are also available, including elapsed time, number of rows fetched, and the size of data fetched or inserted. SQL Logging provides critical information for all server activity, so it can easily be determined who is accessing the system, when the connection occurred, and the work that was 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" topic for more information about the logging facility for SAS Federation Server.

Configuring SQL_Logging

Enable SQL Logging

The **dfs_log_SQL_Logging.xml** configuration file controls the behavior of SQL Logging for SAS Federation Server. This configuration file is located in the **/etc** directory of the server's configuration path. You can set the level of information that each logger captures by specifying a logging level of WARN or TRACE. You can also change logging levels dynamically without stopping the server.

SQL logging is disabled by default. You can enable either full SQL logging, or enable specific transaction loggers to capture information that is suitable for your environment. To enable full SQL logging, open the configuration file and set the value for both of the top-level loggers to TRACE. (The default value for these logs is set to WARN.) To fully enable SQL logging, both of the top-level loggers must be activated. There is no need to change any of the configuration parameters. When you set the two top-level loggers, all transaction logging is enabled by default. In other words, enabling both of the top-level loggers is equivalent to setting all of the transaction loggers to TRACE. It should be noted that server performance might be impacted when SQL logging is fully activated.

You should always set both of these top-level loggers:

```
<logger name="Perf.ARM.FederationServer" additivity="false">
     <level value="TRACE"/>
     <appender-ref ref="ARM"/>
  </logger>

  <logger name="Perf.ARM.SQLServices" additivity="false">
     <level value="TRACE"/>
     <appender-ref ref="ARM"/>
  </logger>
```

Configure Transaction Logging

Following the top-level loggers, in the configuration file, are additional transaction logs that you can use to control logging detail. The information captured by these loggers is enabled when you enable full SQL logging. Therefore, you must disable the two top-

level loggers before configuring the transaction logs. Use the following task to enable individual transaction loggers:

1. Open **dfs_log_SQL_Logging.xml** for editing. This configuration file is located in the **/etc** directory of the server's configuration path.

2. Disable both of the top-level loggers by resetting the level to WARN.

3. Remove the comment marks for each transaction logger that you require, and set the value to TRACE, if it is not already set by default.

To capture minimal information, activate the following logs:

- Perf.ARM.FederationServer.Session.Transaction.SESSION

- Perf.ARM.SQLServices.Connection.Transaction.DBC

- Perf.ARM.SQLServices.Statement.Transaction.SQL

Minimal logging captures overall session, connection, and SQL information. To increase the level of detail that is captured, enable Perf.ARM.SQLServices.Statement.Transaction.CURSOR, and then configure one or more of the following loggers, depending on the needs of your environment:

- Perf.ARM.SQLServices.Statement.BulkOperations

- Perf.ARM.SQLServices.Statement.Execute

- Perf.ARM.SQLServices.Statement.Fetch

- Perf.ARM.SQLServices.Statement.FetchScroll

- Perf.ARM.SQLServices.Statement.Prepare

- Perf.ARM.SQLServices.Statement.SetPos

See the ARM Transactions table for a description of each of these transactions.

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 **Logging** tab of the Federation Server Properties dialog box. When SQL Logging is modified in this manner, 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.

Defining an Application Name

By default the application name (client) name is defined in the federation server configuration file and passed to SQL Logging configuration as **value="X{App.Name}"**. When a SAS Federation Server connection string has been configured with a particular client name, you can define the name in SQL Logging by modifying the following value in the configuration file:

```
<param name="column" value="X{App.Name}" />
```

and replacing App.Name with Client.AppName:

```
<param name="column" value="X{Client.AppName}" />
```

See "Federation Server (FEDSVR) Driver Reference" for information about the APPLICATION NAME connection string option.

Configuring a Third 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 is shipped 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.

1. 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
```

2. Modify the Connection String: Locate the "connection string" parameter in dfs_log_SQL_logging.xml. Modify the connection string to reflect the values for your SQL Logging database. Here is an 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" in this guide.

3. Create the Table: Locate the CREATE TABLE statement in dfs_log_SQL_Logging.xml. At the end of the CREATE TABLE statement, update **'IN <name of your table space>'** to reflect the name of the tablespace for your setup.

```
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>"/>
```

4. Modify the Data Types: Edit dfs_log_SQL_logging.xml to 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" .

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

ARM Transactions

SQL Logging uses Application Response Measurement (ARM) for transaction logging. The table below shows the ARM transactions that are captured in SAS Federation Server. When SQL Logging is enabled, information in each of the transactions is captured.

Table 4.1 *SQL Logging ARM Transactions and Namespaces*

Name	Type	Associated Namespace
SESSION	Session transaction	Perf.ARM.FederationServer.Session.Transaction.SESSION This is a session transaction. A session transaction starts when a user initiates a server session.
DBC	Database connection	Perf.ARM.SQLServices.Connection.Transaction.DBC This is a 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	Database transaction	Perf.ARM.SQLServices.Connection.Transaction.DBTRAN This is an 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	Perf.ARM.SQLServices.Statement.Transaction.SQL This is an 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	SQL Statement	Perf.ARM.SQLServices.Statement.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	SQL Statement	Perf.ARM.SQLServices.Statement.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	SQL Statement	Perf.ARM.SQLServices.Statement.Transaction.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.

Name	Type	Associated Namespace
Fetch	SQL Statement	Perf.ARM.SQLServices.Statement.Fetch
		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.
Fetch Scroll	SQL Statement	Perf.ARM.SQLServices.Statement.FetchScroll
		See Fetch.
SetPos	SQL Statement	Perf.ARM.SQLServices.Statement.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.
BulkOps	SQL Statement	Perf.ARM.SQLServices.Statement.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 return only 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 allows SELECT or DELETE privileges for the active SQL Logging table.

- 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 for which the FedSQL dialect is 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 whether records need to be archived or deleted. Use the following statement to calculate the number 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 4.2   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 name of the logical server registered in SAS Metadata Server.
ARM_NAMESPACE	VARCHAR(256)	The logger name (namespace) of the logging event.
AUTHORIZATION_ID	VARCHAR(128)	UUID for the authenticated user.
AUTHORIZATION_NAME	VARCHAR(128)	The user name from SAS Metadata Server.
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.

Column	Data Type	Description
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, such as 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.
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.

Column	Data Type	Description
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 filename 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.
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 can truncate if the character limit is exceeded. For example, Oracle has a VARCHAR limit of 4000 while SQL Server is 8000.
STATEMENT_STATE	VARCHAR(15)	The state of the SQL statement, such as S0, S1 and S2.
STATEMENT_TEXT	VARCHAR(15000)	The text of the SQL statement. *Note:* The plan value can truncate if the character limit is exceeded. For example, the VARCHAR limit for Oracle is 4000 while SQL Server is 8000.

Column	Data Type	Description
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 62.
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.
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 transact-ion 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 has minimum impact on SAS Federation Server performance, especially during periods of heavy processing.

Using Maximum Buffered Events (MaxBufferedEvents)

The maximum buffered events option 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 onto 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 Third Party DBMS for SQL Logging".

Server Logging Configuration

Introduction

The SAS logging facility is a framework that categorizes and filters log messages in SAS server and SAS programming environments, and writes log messages to various output devices. In the server environment, the logging facility logs messages based on predefined message categories, such as Admin for administrative messages, App for application messages, and Perf for performance messages. The logging facility also enables messages to be filtered based on the following thresholds: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.

The `dfs_log.xml` configuration file controls the destination, contents, and formats of the logging facility log for SAS Federation Server. 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, and server start-up command.

Note: For DS2 logging, see "DS2 Loggers" in the *SAS DS2 Language Reference*.

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.
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 run-time and interface events.
	Debug	Method call and return events.
	Trace	Method parameters.

Logger Name	Logging Level	Description
App.Session	Info	Session object run-time and interface events.
	Debug	Method call and return events.
	Trace	Method parameters.
App.Connection	Info	Connection object run-time and interface events
	Debug	Method call and return events.
	Trace	Method parameters.
App.Statement	Info	Statement object run-time 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 run-time events relating to user identity management and access control logic and enforcement decisions.
Logging	Error	SAS logging facility configuration and run-time events.

Logger Name	Logging Level	Description
Logging.Appender	Error	Appender-specific configuration and run-time events.
Logging.Appender.DB	Error	DB Appender-specific events (used in SQL Logging).
Cradle	Info	General server process framework services, start-up and termination events.
IOM	Info	Processes log events for servers that use the Integrated Object Model (IOM). The IOM interface provides access to SAS Foundation 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.

SQL Loggers

Reserved Loggers for SQL

The following loggers, which are unique to SAS Federation Server, are based on the Audit and App loggers referenced above.

Logger	Logging Level	Description
Audit.SQLFPkg.*package-name*	Debug Trace	Used to log security-related events.
App.SQLFPkg.*package-name*	Debug Trace	Used to log API, logic run time events.

SQL Function Loggers

The following table reflects the Audit and App loggers for SQL functions that are specific to the SAS Federation Server system catalog:

Package-name	Function	Logger
RLS (Row-level security)	SYSCAT.RLS	Audit.SQLFPkg.RLS
		App.SQLFPkg.RLS
DM (Data masking)	SYSCAT.DM	Audit.SQLFPkg.DM
		App.SQLFPkg.DM
UTL (Utilities)	SYSCAT.UTL	Audit.SQLFPkg.UTL
		App.SQLFPkg.UTL

You can use the App.SQLFPkg.UTL logger to verify results after running PROC ASEXPORT during a migration. The log reflects auth IDs that were not properly mapped to an auth ID in SAS Metadata Server. See the *SAS Federation Server 4.2 Migration Guide* for information.

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. The SQL function loggers use DEBUG and TRACE only.

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 configuration 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 SAS 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.

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 works with SAS Metadata Server to perform authentication for users and groups. When a user connects to SAS Federation Server to access data, the user's authenticating credentials are passed to the SAS Metadata Server for validation. Once the credentials are validated, the SAS Metadata Server will identify the user based on the submitted credentials. SAS Federation Server can then make requests to the SAS Metadata Server for information about the user, including logins and group membership. If a user is authenticated but cannot be identified in the SAS Metadata Server, that user becomes a member of the PUBLIC group. All users that are identified in the SAS Metadata Server are members of the SASUSERS group. If a user is designated as SYSTEM or ADMINISTRATOR, their personal credentials are used to authenticate.

It is assumed that the SAS Metadata Server has been installed, configured and populated with user and group information before running SAS Federation Server. For information about adding users and groups, see the *SAS Management Console: Guide to Users and Permissions*.

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. 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 apply to all administration DDL. That is, 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.

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, the 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.

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 these general types:

- Administration permissions
- DS2 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.
EXECUTE	Allows a user or group to execute a DS2 package or method. This authorization is set on the server, data service, catalog, schema, and DS2 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.

Permission	Description
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.
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. [1,2,3]
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. Users with the CREATE CACHE authorization for a server object also have authorization to purge a cache with the PURGE CACHE statement.
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.

[3] ALTER VIEW is required to execute the DESCRIBE VIEW FedSQL statement. See the *SAS FedSQL Reference Guide* for additional information.

Permissions are categorized in the following table.

Administration Permissions	ADMINISTER
	CONNECT
	CREATE DSN
	CREATE CACHE
	ALTER CACHE
	CREATE TABLESPACE
	TRACE
DS2 Permissions	EXECUTE
SQL Permissions	SELECT
	I[DATE
	INSERT
	DELETE
	REFERENCES
	ALTER TABLE
	ALTER VIEW
	DROP TABLE
	DROP VIEW
	CREATE TABLE
	CREATE VIEW

The enforcement of SQL privileges 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 permissions are defined in SAS Federation Server for the user. When the setting is enabled, SAS Federation Server privilege definitions are enforced for the user on that connection.

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

Object Privileges

Object Privilege Inheritance

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

Figure 5.1 *SAS Federation Server Privilege Inheritance*
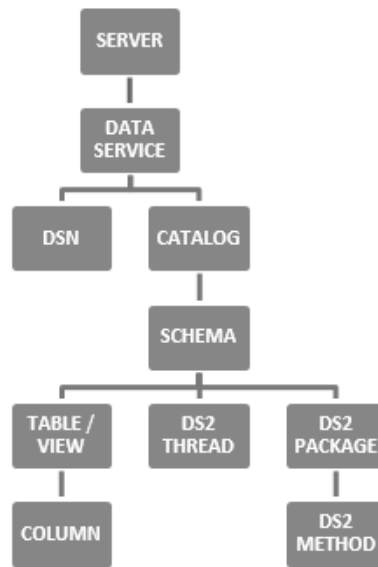
Where privileges on the server are inherited by the data service, privileges on the data service are inherited by the DSN and catalog. Privileges on the catalog are inherited by the schema, table (view), and column. This inheritance hierarchy allows an administrator to set general security rules on higher level objects, and then only set exceptions on the more specific (subordinate) objects.

Figure 5.2 *SAS Federation Server Container Object Inheritance*
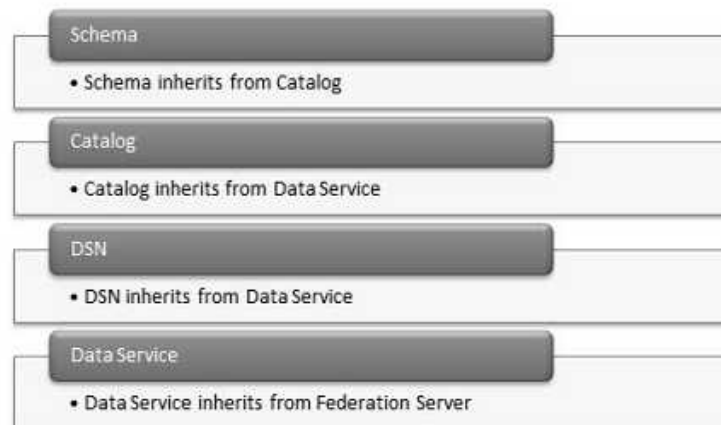
Example:

There is a group called SALES_GROUP whose members are allowed to select most objects in the SALES_DATA data service. An administrator assigns 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, all the schemas of those catalogs, and all the associated tables and views. There is a stipulation that the SALES_GROUP is restricted from viewing any data in a single catalog of the SALES_DATA data service called EXECUTIVE_DATA. To satisfy the requirement, an administrator could then deny SELECT privilege to the SALES_GROUP on that particular catalog. As a result, members of the SALES_GROUP are not 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 exceptions are set on subordinate objects. This minimizes the number of privileges that an administrator must establish, resulting in a reduction of administration overhead. An administrator can request information about privileges held by any user or group for any object in the server hierarchy, 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 **Yes**. The blank fields indicate that there is no privilege inheritance for the object. Inheritance runs from right to left.

Object / Privilege	Column	DS2 Method	Row	Table/ View	DS2 Pkg	DS2 Thread	Schema	Catalog	DSN	Data Service	Fed 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
EXECUTE		Yes			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/ ALTER CACHE				Yes			Yes	Yes		Yes	Yes

Object / Privilege	Column	DS2 Method	Row	Table/ View	DS2 Pkg	DS2 Thread	Schema	Catalog	DSN	Data Service	Fed Server
CREATE TABLESPACE							Yes	Yes		Yes	Yes
ADMINISTER											Yes
TRACE											Yes
CREATE DSN [1]										Yes	Yes
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.

Example:

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.

Determining Effective Privileges

You can use information views to determine effective privileges for federation server objects. Use the "PRIVILEGES and EFFECTIVE_PRIVILEGES" to obtain a complete picture, including inheritance, for server, data services, catalogs and schemas. For tables, views, and columns, use the X_OBJECT_PRIVILEGES and X_COLUMN_PRIVILEGES information views. If the query returns any rows for the grantee, you have the complete picture of privileges, including inheritance. No further queries are needed. However, if the query does not return any rows for the grantee, then you should query schema privileges in EFFECTIVE_PRIVILEGES. The results are the

appropriate privileges to use for the object/column. The INHERITED column should always be set to **Y**.

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 to cache privileges. For example:

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

- The default level is 2. See Level Control below.

- 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. This also applies to DS2 threads and methods.

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

To alter a view definition, you must drop the view and re-create it. When this is done, security for the view must be reestablished. This scenario also applies to security on columns and DS2 objects. Security must be reestablished if these objects are dropped and added at a later time.

Row-Level Security

Introduction

Row-level security (RLS) for SAS Federation Server provides additional security on tables and views by restricting data access on a row-by-row basis. Row-level security allows selection of specific rows for a given set of users and groups. Because row-level security applies only to rows that can be selected, its implementation is a function of the SELECT privilege. The SELECT privilege can be granted without restriction, or with a predicate applied. When a predicate is applied, you are implementing row-level security because the predicate now restricts what rows are returned.

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

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

When members of the SALES group select from table T1, the predicate is automatically attached to the table. When BOB, a member of the SALES group, issues the statement, **SELECT * FROM T1**, the query is effectively transformed to **SELECT * FROM T1 WHERE SALES_REGION = 'NORTHEAST'**.

Reference for syntax details about the GRANT statement.

When 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 in 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 issues 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 the SALES group 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 90 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 SASUSERS 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 produces 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, SASUSERS 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*
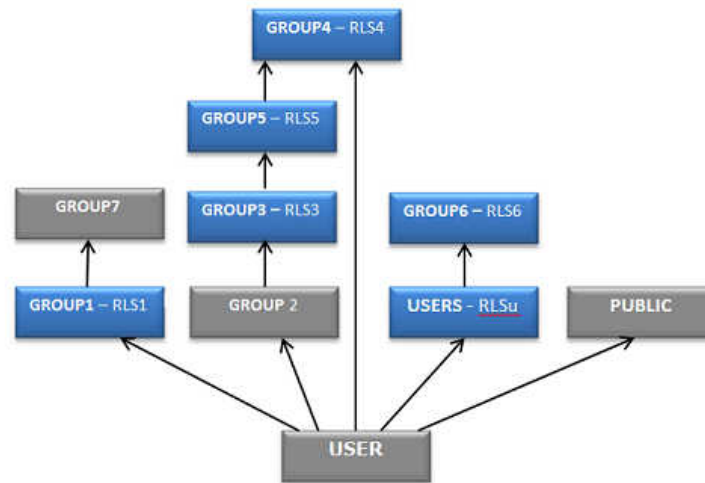
The first predicate, RLS1, is encountered at level 1 in the graph, so the remaining RLS predicates are captured at that level for the current graph, which does not include the secondary graphs, SASUSERS and PUBLIC. The GROUP1 node is marked as visited, and its parent ("member of") associations will not 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 SASUSERS 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 SYSCAT.RLS catalog in the Federation Server Database. The general syntax is **SELECT SYSCAT.RLS.RLS_function('name').** For example:

```
SELECT SYSCAT.RLS.group_id('GROUP1')
```

RLS Functions

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**. Following are the callable functions for row-level security.

auth_id
    Returns the authentication provider identifier for the specified user or group name.

current_user
    Returns the name of the current user.

current_id
> Returns an opaque authentication provider specific user identifier.

domain
> Returns the name of the domain in which the current user is authenticated.

group_id
> Returns the authentication provider group identifier for the specified group name.

login
> Returns the domain-qualified user id that is used to authenticate the current user.

userid
> Returns the SANs domain user id that is used to authenticate the current user. Note that the userid function is similar to the login function, but it is not domain-qualified.

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

is_admin
> Returns TRUE or FALSE indicating if the current user is an administrator.

is_process_user
> Returns TRUE or FALSE indicating if the current user is the process user.

member_of
> Returns TRUE or FALSE indicating if the current user is a member of the specified group.

groups
> Returns a single group name or result set identifying the group memberships of the current user.

The following example uses RLS functions `current_user` and `member_of` to qualify users for SELECT on specific rows in the HR.EMPLOYEES table.

```
grant SELECT on HR.EMPLOYEES to SASUSERS
  where (syscat.rls.current_user() in ("Name","MgrName"))
   or syscat.rls.member_of("Payroll",'DEEP')
```

Authenticated users must meet the following conditions:

1. The name of the current user matches that of the "Name" column (the row pertains to the current user).

2. or the name of the current user matches that of the "MgrName" column (the row pertains to the manager of the current user).

3. or the current user is a direct or indirect (DEEP) member of the "Payroll" group.

RLS Library Reference
Following are the data formats for the row-level security user functions described above.

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 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. 'DEEP' checks for both direct and indirect group membership. 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.

Using the 'ID' and 'DEEP' Options

This topic outlines the behavior of the 'ID' and 'DEEP' options when used with the **member_of** and **groups** RLS functions. Consider the following queries:

SYSCAT.RLS.GROUPS(user_name_or_id [, options])
Returns a group membership list for **user_name_or_id**.

`SYSCAT.RLS.MEMBER_OF(group_name_or_id [, options])`
> Returns TRUE if the current user is a member of the group specified in
> `group_name_or_id`.

The following behavior applies:

- `user_name_or_id` returns a string literal indicating the user name or user ID. If `options` contains 'ID', the argument is treated as a user ID ('6C6C9AD1E2646F0469DD6A3D1874D167') rather than a user name ('USER1').

- `group_name_or_ID` returns a string literal indicating the group name or group ID. If `options` contains 'ID', the argument is treated as a group ID ('78319AD1E2646F0469DD6A3D1874A2F7') rather than a group name ('GROUP1').

- `options` returns a string literal containing 'GROUP', 'ID', or both ('GROUP, ID'). If multiple options are specified, they can be separated in the string by a blank in single quotation marks (' ') or comma in single quotation marks (','). If `options` contains 'ID' the argument is treated as an ID rather than a name. If `options` contains 'DEEP', group membership is checked for direct and indirect membership.

For example, consider that USER1 is a member of GROUP1 and GROUP1 is a member of GROUP2, and 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 use the **GROUPS** function which returns a result set:

`Select * from SYSCAT.RLS.GROUPS('USER1')`
> Returns a result set showing direct group membership for USER1:

```
"GROUP"
'SASUSERS'
'GROUP1'
'PUBLIC'
```

```
Select * from SYSCAT.RLS.GROUPS('USER1', 'DEEP')
```
Using DEEP in the SELECT statement returns a result set showing direct and indirect group membership for USER1:

```
"GROUP"
 'SASUSERS'
 'GROUP1'
 'GROUP2'
 'PUBLIC'
```

Data Masking

Overview

Data masking is a method of hiding sensitive data, or personally identifiable information (PII), within data sources. PII is any data that could potentially identify a specific individual. Any information that can be used to distinguish one person from another, such as government-issued IDs, can be considered PII. The purpose of data masking is to protect original PII data by using a functional substitute in situations where the audience is not privileged to access the original data. The primary focus of data masking is to protect sensitive data while maintaining integrity of the data so that it remains usable.

Data masking with SAS Federation Server uses a set of rules (rule types) and arguments that are run with a system function, **SYSCAT.DM.MASK**. Data masking rules consist of individual rule types that define the specific masking action or algorithm to apply to the data. Data masking rules are as follows:

- ENCRYPT and DECRYPT

- HASH

- TRANC (Transliterated Value)

- RANDOM

- RANDATE (Random Date)

- RANSTR, RANDIG (Random String)

These rules are valid for use in FedSQL queries which are applied over literal values or individual columns to hide PII.

About the SYSCAT.DM.MASK Function

The **SYSCAT.DM.MASK** function accepts defaults configured as package options in addition to the various arguments associated with each rule type. For example, the KEY argument for the ENCRYPT rule defaults to the value configured as the **ENCRYPT_KEY** package option if a key is not specified in the argument. To configure default masking parameters as package options, use the ALTER SERVER DDL command. The following example sets a default, or pre-configured, encryption key used by the ENCRYPT and HASH rule types:

```
ALTER SERVER {options PACKAGE(name 'DM',
    ENCRYPT_KEY '212e8ba6b7f84796a87a985d54277f2f')}
```

Configured parameters revert to a static default value if they are dropped.

Note: Column data encrypted with the static default key cannot be reversed with the **DECRYPT** rule type. If encrypted data will require decryption, you should set a unique encryption key within the data masking argument.

Use the **SYSCAT.DM.MASK** function with the specified rule types and arguments to mask a value containing PII. The rule type argument must be a string constant. Argument names are not case sensitive. Here is the syntax:

```
SYSCAT.DM.MASK( 'rule-type', value [, rule-arguments])
```

Here is the syntax for the rule arguments:

```
[, 'rule-arg-name1', 'rule-arg-value1',
[, 'rule-arg-name2', 'rule-arg-value2', ...]] )
```

Here is an example of data masking that uses the **HASH** rule-type. This rule masks the 'LastName' column in the HR.EMPLOYEES table using an HMAC-MD5 hash and aliases the result as LN:

```
select SYSCAT.DM.MASK('HASH', "LastName",
                       'alg', 'MD5',
                       'key', 'abc123!' ) as "LN"
 from HR.EMPLOYEES
```

Table 5.2 *Data Masking Arguments and Descriptions*

Argument	Description
rule-type	Name of the rule type, for example, ENCRYPT, DECRYPT, or HASH.
value	Specifies the data value to mask, for example, a column reference or other value expression. Value is always the first position following a rule-type in a data masking statement.
rule-arg-name1	Name of the first masking rule argument.
rule-arg-value1	Value of the first masking rule argument.
rule-arg-name2, ...	Name of the second and subsequent masking rule arguments.
rule-arg-value2	Value of the second and subsequent masking rule arguments.

Data Masking Rule Types and Arguments

ENCRYPT / DECRYPT

ENCRYPT masks the values in a column by encrypting or encoding a single value using a symmetric key cipher or simple encoding algorithm. To achieve truly encrypted results, a KEY argument must be specified, or a default **ENCRYPT_KEY** pre-configured in **SYSCAT.DM.MASK**.

DECRYPT unmasks previously encrypted values using a symmetric key cipher or a simple decoding algorithm. To unmask previously encrypted or encoded values, the encryption key must be specified using the KEY argument. DECRYPT will not use the

encryption key configured in **SYSCAT.DM.MASK**. The DECRYPT rule returns NULL for the AES algorithm if the KEY argument is omitted.

Here is the syntax for ENCRYPT and DECRYPT:

```
SYSCAT.DM.MASK( 'ENCRYPT', value [, rule-arguments ] ),
SYSCAT.DM.MASK( 'DECRYPT', value [, rule-arguments ] )
```

Note: Use HASH if uniqueness, reversal, or decryption is not a requirement.

Unique	Results are unique if the input values are unique.
Deterministic	Results are deterministic if DETERMINISTIC YES is specified.
Reversible	Reversal is possible if key is poor quality or divulged.

Arguments for ENCRYPT and DECRYPT

Use the following values to specify a data masking function with ENCRYPT or DECRYPT:

VALUE

Specifies the data value to mask. Value can be a column or other value expression, and always follows the ENCRYPT or DECRYPT rule-type in a masking statement. Here is an example where the social security number column ("SSN") is masked with encryption:

```
select SYSCAT.DM.MASK('ENCRYPT', "SSN",...
```

ALG

Argument type: Algorithm required. Case-insensitive string constant.

Specifies the algorithm name which is one of the following:

AES/FIPS	FIPS 140-2 compliant AES encryption (FIPS 197). Symmetric key block encryption cipher using the AES (Advanced Encryption Standard) algorithm for 128 bit blocks using a 256 bit key and a 64 bit salt value. An encryption key must be specified in order for the algorithm to produce truly encrypted results. Requires installation of DataFlux Secure.
AES	FIPS 140-2 compliant AES encryption (FIPS 197). Symmetric key block encryption cipher using the AES (Advanced Encryption Standard) algorithm for 128 bit blocks using a 256 bit key and a 16 bit salt value. An encryption key must be specified in order for the algorithm to produce truly encrypted results. Requires installation of DataFlux Secure.
BASE64	Base 64 encoding. Not FIPS 140-2 compliant.
SAS001	Alias for BASE64 encoding.
SAS002	SAS Proprietary encoding. Not FIPS 140-2 compliant.
SAS003	Alias for AES.
SAS004	Alias for AES/FIPS.

KEY

Argument type: Case-insensitive string constant. Valid only with AES encrypted data. Required for DECRYPT. Not valid for the SAS002 algorithm.

Symmetric encryption/decryption key. The key defaults to the pre-configured **ENCRYPT_KEY** option in SYSCAT.DM.MASK, or, for **ENCRYPT** only, the static

default when none is configured. You can set a pre-configured encryption key using the ENCRYPT_KEY option in the ALTER SERVER DDL statement.

Note: DECRYPT will not use a pre-configured encryption key. Therefore, you must specify a KEY argument for decryption to work.

DETERMINISTIC

Argument type: Case-insensitive Boolean string constant or 1 or 0.

Use with **ENCRYPT** only.

Boolean string constant values must contain one of {YES, TRUE, ON, 1, NO, FALSE, OFF, 0 }.

Use this option when deterministic output is required. Controls whether encrypted value is DETERMINISTIC. The default value is FALSE.

Exception: Not valid with the SAS002 algorithm, which does not support a custom key.

EXPAND_PREC

Argument type: Case-insensitive Boolean string constant or 1 or 0.

Use with **ENCRYPT** only.

Boolean string constant values must contain one of {YES, TRUE, ON, 1, NO, FALSE, OFF, 0 }.

This option causes the precision of the output value to accommodate the possibility of data bloat when using the specified encryption or encoding method. **EXPAND_PREC** is active by default. If the encrypted value does not fit in the column, this option returns an empty string for **VARCHAR** or **NVARCHAR**, or all-blank for **CHAR** or **NCHAR**.

- an empty string for **VARCHAR** or **NVARCHAR**.

- a completely padded blank for **CHAR** or **NCHAR**.

CASE

Argument type: Case-insensitive string constant.

Use with **ENCRYPT** only.

Controls case in output. Does not apply to the BASE64 algorithm, which produces upper and lower cased characters.

U[PPER]   Use uppercase hexits A-F.

L[OWER]   Use lowercase hexits a-f.

STRIP

Argument type: Case-insensitive string constant.

Use with **ENCRYPT** only.

Specifies whether to strip trailing whitespace characters from the input value prior to encryption. By default, values are not stripped.

Valid values are BLANKS, UNICODESP, UNICODESPACE, ANY, ALL, WS. BLANKS specifies to strip ASCII blank (0x20) characters only. UNICODESP and UNICODESPACE specify to strip Unicode whitespace characters. ANY, ALL, and WS specify to strip Unicode whitespace characters as well as certain format control characters.

HASH

The HASH rule hashes a single value into a fixed-length hash digest or HMAC string and is not reversible. Here is the syntax for HASH:

```
SYSCAT.DM.MASK( 'HASH', value [, rule-arguments ] )
```

Unique	Results might be unique if the input values are unique.
Deterministic	Yes
Reversible	No

Algorithm	Value Type	Return Type
MD5	CHAR or VARCHAR	CHAR(32)
	WCHAR or WVARCHAR	WCHAR(32)
	other	BINARY(16)
SHA256	CHAR or VARCHAR	CHAR(64)
	WCHAR or WVARCHAR	WCHAR(64)
	other	BINARY(32)

Arguments for HASH

Use the following values to specify a HASH data masking function:

VALUE
Specifies the data value to mask. Value can be a column or other value expression, and always follows the HASH rule-type in a masking statement. Here is an example where "LastName" is the masked value.

```
select SYSCAT.DM.MASK('HASH', "LastName",...
```

ALG
Argument type: Case-insensitive string constant.

Required. Specifies the algorithm name:

MD5	Robert Rivest's 128–bit algorithm (1991).
SHA256*	NSA'S 256–bit algorithm (2001). SHA256 is the default.

Note: *SHA256 requires installation of DataFlux Secure software.

CASE
Argument type: Case-insensitive string constant.

Controls case in output. Does not apply to the BASE64 algorithm, which produces upper and lower cased characters.

U[PPER]	Use uppercase hexits A-F.
L[OWER]	Use lowercase hexits a-f.

KEY
Argument type: Case-insensitive string constant.

By specifying a KEY argument, or defaulting to the ENCRYPT_KEY parameter configured in the package, a 'hash message authentication code' (HMAC) that complies with RFC 2104 (http://tools.ietf.org/pdf/rfc2104) is computed. Otherwise, the specified raw digest is computed directly.

TRANC

TRANC masks the values in a column by transliterating characters from the input string to characters in the output string. Ensure that the mapped result contains 'many-to-1' character transliterations so that an inverse transliteration does not determine the original value. The TO and FROM strings are case–sensitive and should be lower cased. Here is the syntax for TRANC:

```
SYSCAT.DM.MASK ('TRANC', "value" ,
          'FROM', 'lower-case string,
          'TO', 'lower-case string' )
```

Unique	Data dependent.
Deterministic	YES
Reversible	Reversal is possible depending on the character mapping provided. Reversal is appropriate when mapping multiple input characters to a single output character.

Here is the syntax for TRANC:

```
SELECT SYSCAT.DM.MASK('TRANC', TABLE.COLUMN."FIELD", 'FROM', 'characters_to_convert',
 'TO', 'converted_characters',
'START', 1, 'LENGTH ', 11) )

'TRANC', '123', 'FROM', '0123456789','TO', 'XXXXXXXXXX',
   'START', 1, 'LENGTH', 3
```

Arguments for TRANC

Use the following values to specify a data masking function with TRANC:

VALUE

Specifies the data value to mask. Value can be a column or other value expression, and always follows the TRANC rule-type in a masking statement.

FROM

Argument type: String

Specifies the characters to convert from.

TO

Argument type: String

Specifies the characters to convert to. The **TO** string must not have more characters than the **FROM** string. Additional **FROM** string characters are mapped to blanks.

START

Argument type: String

Specifies the starting position. The default starting position is 1.

LENGTH

Argument type: Integer

Specifies the transliteration length. The default is the length of entire input string.

RANDOM

RANDOM masks the values in a numeric column which results in a uniformly distributed pseudo-random number.

Unique	Not guaranteed and dependent on the value range.
Deterministic	YES unless NULL constant is specified as the value.
Reversible	Not applicable.

Here is the syntax for RANDOM:

```
SELECT SYSCAT.DM.MASK('RANDOM', '123', 'MIN', 1239, 'MAX', 10000, 'VARY', 10.5,
'KEY', '5c39b18d77d5f297ff92e4942e5522b5')
```

Arguments for RANDOM

Use the following values to specify a RANDOM data masking function:

VALUE

Specifies the data value to mask. Value can be a column or other value expression, and always follows the RANDOM rule-type in a masking statement.

SEED

Argument type: 64-bit signed integer

Initial integer seed. If omitted, a default seed is supplied from the **RANDOM_SEED** package configuration parameter. You can set a RANDOM_SEED using the "ALTER SERVER" .

MIN

Argument type: Numeric

MIN specifies the minimum value and accepts either NULL or NOT NULL values. Either MIN and MAX or VARY is required in the argument.

MAX

Argument type: Numeric

MAX specifies the maximum value and accepts either NULL or NOT NULL values. Either MIN and MAX or VARY is required in the argument.

VARY

Argument type: Numeric

Vary original value by **+/-** variance of the amount. VARY requires a NOT NULL value. Mutually exclusive of MIN and MAX parameters. Therefore, VARY is required if MIN or MAX is not used.

KEY

Argument type: String

Specifies a secret key that is used to produce an HMAC internally from which the pseudo-random result can be computed deterministically based on the value. A quality key is necessary to prevent discovery of the value using a rainbow table attack. KEY is used in conjunction with the input value to compute a cryptographic hash message authentication code (a derived key) from which to compute the pseudo-random output. KEY works with MIN or MAX and VARY.

Note: Use KEY with a non-NULL argument to produce deterministic results. The KEY value is combined with the value passed to the function to produce deterministic values. Use SEED with a NULL argument to produce non-deterministic results. Either KEY or SEED is used, but not both. If both are specified one will be ignored, depending on whether the value is NULL.

RANDATE

RANDATE masks the values in a date column by replacing them with pseudo-random date values.

Unique	Not guaranteed and dependent on the date range.
Deterministic	YES unless NULL constant is specified as the value.
Reversible	No.

Arguments for RANDATE

Use the following values to specify a data masking function with RANDATE:

VALUE

Specifies the data value to mask. Value can be a column or other value expression, and always follows the RANDATE rule-type in a masking statement.

SEED

Argument type: 64-bit signed integer

Initial integer seed. If omitted, a default seed is supplied from the **RANDOM_SEED** package configuration parameter. You can set a RANDOM_SEED using the "ALTER SERVER" .

VARY

Argument type: Numeric

Vary original value by **+/-** variance of the amount. Mutually exclusive of MIN and MAX parameters.

U[NITS]

Argument type: String

Specifies the variance units. Numeric types are treated as a SAS date value. The possible units are:

```
DAY, D
WEEK, WK, W
MONTH, MO
YEAR, YR, Y
HOUR, HR, H
MINUTE, MIN, M
SECOND, SEC, S
```

Note: Random date variances include DATE, TIME, and TIMESTAMP column types. The default unit for TIME or TIMESTAMP data types is HOUR, and MONTH for others.

KEY

Argument type: String

Specifies a secret key that is used to produce an HMAC internally from which the pseudo-random result can be computed deterministically based on the value. A quality key is necessary to prevent discovery of the value using a rainbow table attack. KEY is used in conjunction with the input value to compute a cryptographic hash message authentication code (a derived key) from which to compute the pseudo-random output.

Note: Use KEY with a non-NULL argument to produce deterministic results. The KEY value is combined with the value passed to the function to produce deterministic values. Use SEED with a NULL argument to produce non-

deterministic results. Either KEY or SEED is used, but not both. If both are specified one will be ignored, depending on whether the value is NULL.

RANSTR

RANSTR masks the values in a column by replacing with random strings. Strings are generated by an algorithm that uses characters from the source string in the generation process, adding padding characters if necessary. Padding is placed to the left of the string unless RIGHT is specified. The minimum number of non-pad characters is MINPREC, and the maximum is MAXPREC. The value passed to RANSTR is used only to ensure deterministic results.

Use the following values to define a data masking function with RANSTR:

Type	WCHAR(MAXPREC) if padding is off. WVARCHAR(MAXPREC) if padding is on.
Unique	Not guaranteed but more so as MINPREC is increased.
Deterministic	YES unless NULL constant is specified as the value.
Reversible	Not applicable.

Here is the syntax for RANSTR:

```
SELECT SYSCAT.DM.MASK('RANSTR', NULL, 'MINPREC', 10, 'SOURCE', 'Hello World')
```

Arguments for RANSTR

Use the following values to specify a data masking function using RANSTR:

VALUE

Specifies the data value to mask. Value can be a column or other value expression. Value always follows the RANSTR rule-type in a masking statement.

SEED

Argument type: 64-bit signed integer

Specifies the initial seed.

MINPREC

Argument type: Integer

Specifies the minimum string precision. The default minimum precision is 0.

MAXPREC

Argument type: Integer

Specifies the maximum string precision. By default, **MAXPREC=MINPREC**.

SOURCE

Argument type: String

Specifies the characters from which to create the random string. The source value can be a column reference.

PAD

Argument type: String

Specifies a PAD character or NULL. If NULL is specified, no padding is added to the generated string. PAD uses **' '** as the default character if a character is not specified.

RIGHT

Argument type: Boolean

Specifies that pad is on the right side of the generated string.

KEY
Argument type: String

Specifies a secret key that is used to produce an HMAC internally from which the pseudo-random result can be computed deterministically based on the value. A quality key is necessary to prevent discovery of the value using a rainbow table attack. KEY is used in conjunction with the input value to compute a cryptographic hash message authentication code (a derived key) from which to compute the pseudo-random output.

Note: Use KEY with a non-NULL argument to produce deterministic results. The KEY value is combined with the value passed to the function to produce deterministic values. Use SEED with a NULL argument to produce non-deterministic results. Either KEY or SEED is used, but not both. If both are specified one will be ignored, depending on whether the value is NULL.

RANDIG

RANDIG masks the numeric values in a column by replacing digits them with strings of random digits. Strings are generated by an algorithm that uses digits derived from the base number system of the source value, adding padding digits if necessary. RANDIG is an alias of RANSTR with the following constraints:

*   Padding is always to the left of digits.

*   Pad character defaults to '0' for bases other than 64, and '' for base 64.

*   SOURCE is implicit and derived from the value of BASE.

Here is the syntax for RANDIG:

```
SELECT SYSCAT.DM.MASK ('RANDIG', NULL, 'SEED', 100, 'BASE', 2)
```

Arguments for RANDIG

Use the following values to specify a data masking function with RANDIG.

RANDIG Only Parameters:

VALUE
Specifies the data value to mask. Value can be a column or other value expression, and always follows the RANDIG rule-type in a masking statement.

SEED
Argument type: 64-bit signed integer

Specifies the initial seed. Use with a NULL argument to produce non-deterministic results.

MINPREC
Argument type: Integer

Specifies the minimum string precision. The default minimum precision is 0.

MAXPREC
Argument type: Integer

Specifies the maximum string precision. By default, **MAXPREC=MINPREC**.

BASE
Argument type: Integer

2   Binary

8   Octal

10   Decimal (default)

16   Hexadecimal

64   Base 64

CASE
Argument type: String

U[PPER]   Use uppercase hexits A-F (default).

L[OWER]   Use lowercase hexits a-f.

The CASE option is ignored for bases other than 16.

KEY
Argument type: String

Specifies a secret key that is used to produce an HMAC internally from which the pseudo-random result can be computed deterministically based on the value. A quality key is necessary to prevent discovery of the value using a rainbow table attack. KEY is used in conjunction with the input value to compute a cryptographic hash message authentication code (a derived key) from which to compute the pseudo-random output.

Note: Use KEY with a non-NULL argument to produce deterministic results. The KEY value is combined with the value passed to the function to produce deterministic values. Use SEED with a NULL argument to produce non-deterministic results. Either KEY or SEED is used, but not both. If both are specified one will be ignored, depending on whether the value is NULL.

Server Encryption

Introduction

SAS Federation Server supports two methods of encryption strength: SAS Proprietary Encryption and encryption using DataFlux Secure. SAS Federation Server Manager uses SAS/SECURE for encryption.

SAS Proprietary Encryption

SAS Proprietary Encryption 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 Encryption algorithm is strong enough to protect your data from casual viewing. SAS Proprietary Encryption provides a medium level of security. SAS/SECURE and SSL provide a high level of security. See *Encryption in SAS* for additional information.

DataFlux Secure

Overview
DataFlux Secure is an add-on product that provides industry encryption capabilities in addition to the SAS Proprietary Encryption 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.

Specifying the Encryption Method
SAS Proprietary Encryption (SASProprietary) is the default encryption for SAS Federation Server. You can also decide how much data is encrypted in communication between a client and SAS Federation Server. Encryption is specified on the federation server object located in SAS Metadata Server. Follow these steps to change the encryption level using SAS Management Console. Use the following procedure to specify or change the encryption level for SAS Federation Server.

1. Using SAS Management Console, locate your federation server object by expanding **Environment Management** ⇨ **Server Manager** ⇨ **Federation Server -** *hostname* **- logical server**.

2. Expand the logical server entry and select the server definition that you wish to change encryption for. The **Connections** tab displays the current connections defined for the selected server.

3. On the **Connections** tab, select **Bridge connection** and right-click. Select **Properties** from the drop-down menu.

4. Select the **Options** tab and select **Advanced Options**.

5. Select the **Encryption** tab and select an option from the **Server encryption algorithm** list menu.

6. Click **OK** to exit the Advanced Options dialog box, and click **OK** to close connection properties.

7. Restart SAS Federation Server to update the server encryption algorithm.

Chapter 6
SAS Languages

Overview

SAS Federation Server provides support for the FedSQL and DS2 languages through the use of a 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

About 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.

FedSQL is a requirement for many functions on SAS Federation Server such as data federation, cached views, row-level security, and data quality.

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

About the FedSQL Language Driver

The FedSQL language driver supports the FedSQL dialect. 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 includes an SQL processor, which supports the FedSQL dialect. The primary function of the FedSQL driver is to support federation of data sources. if an SQL submission is requesting data from DB2 to be joined with data from Oracle, the SQL processor requests the data from the data sources and then performs the join in SAS Federation Server. The FedSQL driver supports the FedSQL dialect over any data source. For example, if the SQL request is from a single data source that does not support a particular SQL function, the FedSQL processor guarantees implementation of the request.

See the "FedSQL Driver Reference" for a complete list of connection options.

Invoking the FedSQL Dialect

To invoke FedSQL, configure a DSN using these scenarios::

* Specify the dialect in a DSN using CREATE DSN with the LANG connection option. Federated DSNs require that the FedSQL dialect be set. For BASE DSNs, the dialect defaults to FedSQL.

  ```
  CREATE DSN "DSN1" UNDER BASE DESCRIPTION 'creating DSN1' NOPROMPT
  'DRIVER=BASE;CATALOG="catalog1_BASE";SCHEMA="schema1"' {OPTIONS (LANG YES)}
  ```

  Note: You can execute the only syntax supported by the language based on the dialect specified in the LANG option. You cannot execute FedSQL with DS2 dialect, and you cannot execute DS2 statements using the FedSQL dialect.

* Specify "Invoking the FedSQL Dialect" by setting SECURITY to YES on a DSN. When SECURITY is set to YES, FedSQL is automatically set to YES.

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

See CREATE DSN DDL for details about DSN options.

Federation Server SQL Authorization Enforcement

When Federation Server SQL Authorization Enforcement is enabled, the FedSQL driver is enabled, 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 the user is granted all privileges regardless of what the user has been granted or denied.

When Federation Server SQL Authorization Enforcement is disabled in SAS Federation Server Manager, an administrator has the option to choose native dialect, which is the dialect of the underlying data source. For example, if you are connected to Oracle, then the native dialect would be SQL supported by Oracle.

To disable Federation Server SQL Authorization Enforcement using administration DDL, include **FEDSQL=NO** in the CREATE DSN statement to default to native dialect.

DS2

About DS2

DS2 is a SAS proprietary programming language that is used for advanced data manipulation. DS2 provides capabilities not available through SQL, such as scoring models. In addition, you can use DS2 code to run data quality functions on SAS Federation Server. DS2 is included with Base SAS and intersects with the SAS DATA step.

To invoke DS2, you must configure a DSN that uses the DS2 dialect and grant users CONNECT permission to the DSN. In addition, users must have EXECUTE permissions on DS2 objects, such as packages and threads, before any SQL functions can be run against them.

DS2 objects inherit privileges from the server in the following order:

- SERVER
- (DATA) SERVICE
- CATALOG
- SCHEMA
- PACKAGE
- FUNCTION

For more information about DS2, see the *SAS DS2 Language Reference*.

Viewing DS2 Package Contents

To view the contents of DS2 programs, use the DESCRIBE PACKAGE or DESCRIBE THREAD command. Only administrators or schema owners can run these commands, unless the PL Source Management Security option is set to FALSE.

```
DESCRIBE PACKAGE DS2-program-name
DESCRIBE THREAD DS2-program-name
```

User Permissions

Users require EXECUTE permissions on DS2 objects, so they can submit a DS2 code stream or manage packages and threads. Use the GRANT, DENY, or REVOKE DDL statements to establish permissions for a user or group.

Here are examples that grant, deny, and revoke the EXECUTE permission on a DS2 package or thread:

```
GRANT EXECUTE on package basecat.basetest.pkgGrade to "user-name"
DENY EXECUTE on package basecat.basetest.pkgGrade to "user-name"
REVOKE EXECUTE on package basecat.basetest.pkgGrade from "user-name"
```

Only administrators and schema owners can CREATE or DROP a DS2 object.

The table below summarizes permissions for DS2 objects.

Object Permission		DS2 Data Program	DS2 Thread	DS2 Package	Method in a DS2 Package
CREATE		Not applicable	Connect DS2 DSN / EXECUTE on server (when connection string is used). Schema owner	Connect DS2 DSN / EXECUTE on server (when connection string is used). Schema owner	Created along with a DS2 package.
DROP		Not applicable	Connect DS2 DSN / EXECUTE on server (when connection string is used). Schema owner	Connect DS2 DSN / EXECUTE on server (when connection string is used). Schema owner	Dropped along with a DS2 package.
EXECUTE	SQL/Call statement	Not applicable	Not applicable	Not applicable	Connect DS2 DSN with FEDSQL dialect. EXECUTE on the DS2 method.
	DS2 program	Connect DS2 DSN / EXECUTE on server (when connection string is used).	Connect DS2 DSN / EXECUTE on server (when connection string is used). EXECUTE on the DS2 thread.	Connect DS2 DSN / EXECUTE on server (when connection string is used). EXECUTE on the DS2 thread.	Connect DS2 DSN / EXECUTE on server (when connection string is used). EXECUTE on the package that contains the method.

Administrators have the required privileges to run or manage DS2 objects for all of the cases described above.

Note: Regarding input/output tables: If a DS2 program reads data from an input data set, SELECT privilege is required on the input table. If a DS2 program creates and output data set, then CREATE, INSERT, SELECT, and DROP are required on the output table.

Securing DS2 Objects

DS2 objects that must be secured include package functions and threads. Typically, a function is invoked through a CALL statement, a FedSQL function, or referenced from a DS2 data program. Functions and objects referenced from a DS2 program are secured with the EXECUTE privilege. To grant or deny the EXECUTE privilege on a DS2 package or thread, use the following DDL statements:

```
GRANT EXECUTE on package package-name to username
```

```
DENY EXEUCTE on package package-name to username
```

When placing the EXECUTE privilege on a function, and specifying names in the statement, only three levels of naming are allowed. Therefore, place quotation marks around the **"packagename.functionname"** so that it is treated as one object name. Here are some examples:

```
GRANT/DENY EXECUTE on FUNCTION basecat.basetest."pkgGrade.compute" to username
REVOKE EXECUTE on FUNCTION basecat.basetest."pkgGrade.compute" from username
```

You can also place the EXECUTE privilege on specific DS2 object containers, which are inherited by other DS2 objects:

```
GRANT EXECUTE on SERVICE BASE to PUBLIC;
DENY EXECUTE on CATALOG basecat to PUBLIC;
GRANT EXECUTE on SCHEMA basecat.basetest to public;
```

Invoking DS2

Configure a DS2 DSN

You can configure a standard DSN or a federated DSN with the DS2 dialect. Use the CREATE DSN statement with the LANG option to invoke the DS2 dialect, [for example, **{OPTIONS (LANG DS2)}**]. Here is the syntax:

```
CREATE DSN dsn-name under BASE noprompt 'catalog="catalog-name";
schema="schema-name"' {OPTIONS ( LANG DS2 )};
```

Here are some examples of a standard DSN using the DS2 dialect:

```
CREATE DSN test UNDER data-service-name NOPROMPT
'CATALOG=LD_CAT1_BASE;SCHEMA=(NAME=LD_SCHEMA1_BASE)' {OPTIONS (LANG DS2,
SECURITY YES, CREDENTIALS_SEARCH_ORDER(SHARED))}
```

A federated DS2 DSN can include DS2, FedSQL, or native child DSNs. Here is an example of a federated DSN that uses DS2:

```
CREATE DSN federated-DS2 UNDER BASE {OPTIONS ( LANG DS2 )} ADD (DSN1, DSN2, DSN3)
```

You can also configure DSNs with the DS2 dialect using SAS Federation Server Manager. Create a new standard or federated DSN and select the **DS2** language at the Syntax dialog box.

Figure 6.1   *DS2 DSN, SAS Federation Server Manager*
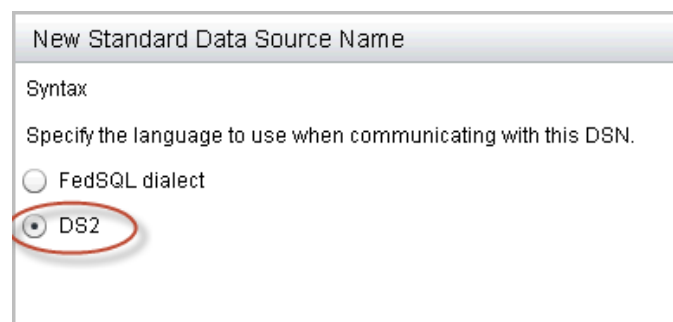
See the *SAS Federation Server Manager: User's Guide* for detailed instructions to create DSNs.

Execute the Code

After creating the DSN with the DS2 dialect, execute the code using the Console in SAS Federation Server Manager.

1. Open the Console in Federation Server Manager and perform the following actions:

    a. Using the **Server** list menu, select the federation server that contains the Base DSN for DS2.

    b. Using the **Connection** list menu, select the Base DSN for DS2.

    c. Using the list menu at **Select connection language**, select **DS2** and click **OK**.

2. Enter, or copy, your statement in the Console and click **Submit**.

```
data outtable(overwrite=yes);
dcl double x;
method run();
x=1;
output;
end;
```

Chapter 7
Data Source Access

Working with Data Services

Overview of Data Services

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.

Data services contain information that identifies the location of tables residing in your
data source. 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
unique identification of each data source when performing heterogeneous operations.

Data services that require logins must be associated with a domain in SAS Metadata
Server. When users connect to the data source through a data source name (DSN), 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
SAS Metadata 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 data source and you can assign privileges that control
user access to the data. However, relational databases provide authorization that limits
the operations that can be performed on the data. SAS Federation Server respects
authorizations that are defined and enforced on a third-party database. Authorizations

defined on a third-party database overrule permissions and privileges that are set on SAS Federation Server.

Creating a Data Service

Overview

You can use administration DDL or SAS Federation Server Manager to create a data service. A new DSN is automatically generated each time a new data service is created. This is a standard DSN given the same name as the data service. However, if the DSN name already exists on the server, it will not be created. If the data service is renamed, the DSN name remains unchanged.

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 fails. 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 server's 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 in SAS Metadata Server.

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 SAS Metadata Server for it to be accessible when creating the data service with 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 241 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]
```

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 252 statement to provide a mapped name for the native name.

Driver Search Order

If specifying more than one driver when creating a data service, the first driver listed in the statement is used as the default driver. Here is an example data service definition:

```
CREATE DATA SERVICE ORACLE_TEST TYPE ORACLE CATALOG ORACLE_TEST DOMAIN ORA1
  {OPTIONS CONOPTS (DRIVER ORACLE, PATH TKTSORA ), CONOPTS (DRIVER ODBC,
      ODBC_DSN TRAFFIC)}
```

In this case, the native ORACLE driver is the default. To change the default driver, you must first drop the drivers from the data service using ALTER SERVICE DDL. After dropping the drivers, add the drivers to the data service putting the default driver first. In the following example, two drivers, ORACLE and ODBC, are dropped from the ORACLE_TEST data service:

```
ALTER DATA SERVICE ORACLE_TEST {OPTIONS DROP CONOPTS(DRIVER ORACLE), DROP
CONOPTS(DRIVER ODBC)}
```

After the drivers are dropped, add the drivers again, specifying ODBC first so that it becomes the default driver:

```
ALTER DATA SERVICE ORACLE_TEST {OPTIONS ADD CONOPTS ( DRIVER ODBC, ODBC_DSN
TRAFFIC ), ADD CONOPTS ( DRIVER ORACLE, PATH TKTSORA ) }
```

Working with DSNs

Overview of DSNs

DSNs are resources that provide connection information for data sources accessed through SAS Federation Server. The administrator assigns permissions that determine how users connect to the data. For example, to connect to a data source, a user must be granted CONNECT permission on SAS Federation Server, a specific data service, or a specific DSN.

A DSN references a specific data source 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 permission is required to create a DSN. You can configure DSNs using Administration DDL statements or by using SAS Federation Server Manager. All DSNs must be associated with a data service, except for federated DSNs which are objects parented by the federation server.

DSN Types

Standard DSN

A standard DSN is a single-service DSN created for a 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.

Federated DSN

A federated DSN is a collection of one or more 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.

System DSNs

These system DSNs are created during installation of SAS Federation Server:

ADMIN DSN

The **ADMIN DSN** is created at server start up 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 264. 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` permission to the ADMIN DSN. SAS Federation Server automatically checks user privileges when administration SQL is submitted. Users can submit administration SQL for which they have privileges, such as selecting against Information Views. Some administration SQL can be executed by the server administrator only. See the *SAS FedSQL Reference Guide* for details.

SQL_LOG DSN

A **SQL_LOG DSN** and data service are created when SQL Logging is enabled on SAS Federation Server. At that time the server creates an EVENTS table for the purpose of capturing server activity that reflects information about SQL statements submitted by connected users. Additional information can be found in "SQL Logging" on page 60.

Note: The CONNECT permission is not assigned to a DSN by default, and must be granted by the administrator or by the DSN owner, to users or groups that are connecting to data sources.

Permissions for DSN

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

The permissions for standard and federated DSNs are:

• CREATE DSN

• ALTER or DROP DSN

• CONNECT

CREATE DSN

To create a DSN, one of the following conditions must be met:

• The user is a system user.

• The user is an administrator of the server.

• The user has the CREATE DSN permission on the server. Note that this is the only way that a user who is not an administrator can create a federated DSN.

- The user has the CREATE DSN permission on the data service, and the user is creating a standard DSN.

ALTER/DROP DSN

To alter or drop a DSN, one of the following conditions must be met:

- The user is a system user.

- The user is an administrator of the server.

- The user is the owner of the DSN.

CONNECT

A user must have CONNECT permission to establish connection to a DSN. This permission is effective from the user object, inherited through the hierarchy, or acquired through group permissions. For a standard DSN, the CONNECT permission must be on (in order of inheritance):

- The DSN,

- The parent data service of the DSN, or

- SAS Federation Server.

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

- The DSN, or

- SAS Federation Server.

Permissions granted on a federated DSN override any permissions that exist for child DSNs that are contained within the federated DSN. If a user has CONNECT permission on a federated DSN, permissions on any of the child DSNs contained within (standard or federated) are ignored, even if the user is explicitly denied CONNECT on any of the child DSNs

For additional information about permission assignment, see the topic about Permissions on page 80.

Configuring DSNs

Creating a DSN with Administration DDL

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 247 .

Standard DSN

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

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

Here is a DSN that uses a GROUP login:

```
CREATE DSN "dsn-name"
   UNDER "data-service"
   CONNECT 'DRIVER=Oracle;GROUP=group-name' {OPTIONS CSO PERSONAL}
   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. When creating a federated DSN, ensure that the child DSNs are not pointing to the same catalog, as this might result in a catalog conflict error. Catalog names must be unique within a connection. Here is the syntax for creating a federated DSN:

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

DSN Login Credentials

If data services require credentials, a 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.

When using SAS Federation Server Manager, an administrator can specify 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). Therefore, if an administrator has both a personal and a shared login, the personal login is used. If an administrator does not have a personal login, but has multiple shared logins available, the connection might be disallowed.

Credentials Search Order (CSO) for DSN Connections

Connections made with 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).

At connection request, SAS Federation Server attempts to select a user ID and password for each data service connection based on the associated domain:

- PERSONAL means the server attempts to select credentials directly owned by the user. This includes group-owned logins when the user does not own a login in the domain of the service to which the DSN is associated.

- SHARED means the server attempts to select credentials from a shared login of which the user is a consumer. Credentials are extracted on behalf of the user that is using the shared login.

  - 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 if the search order is not CSO(SHARED), the connection is still attempted. If credentials are specified on the connection string, those will be used first. If credentials are not supplied, the server attempts to find shared logins for the user. If shared login credentials are not found, the server attempts to use personal credentials. If personal credentials are not found, the connection fails.

Enabling Federation Server SQL Authorization

When Federation Server SQL Authorization 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

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 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)}
```

FedSQL Pass-Through Facility

The FedSQL pass-through facility enables you to connect to a data source and send SQL statements directly to that data source for execution. This facility also enables you to use the syntax of your data source, and it supports any non-ANSI standard SQL that is supported by your data source. SAS Federation Server supports FedSQL pass-through with the use of personal credentials for the connection. Shared logins are not allowed with FedSQL pass-through. See the *FedSQL Reference Guide* for additional information about the pass-through facility.

Shared Logins: Best Practices

Shared logins consist primarily of a login and a domain to share, and the consumers who use that login. As a best practice, the consumers will typically list one or more groups. However, a conflict can arise when a particular user is in a consumer group, either directly or indirectly, of multiple shared logins for the same domain. The following scenarios outline shared login conflicts and their resolution.

Scenario 1: Application Users

In the following scenario, an application exists which requires the use of a particular set of database credentials to access protected data.

In this example, an HR application has data content stored in Oracle and DB2.

Use the following procedure 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 with the CONNECT option (**CONNECT 'group=HR_USERs'**), or in a connection string that specifies a DSN (**group=HR_USERS;dsn=HR1**). 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 specific operations, for example, SELECT versus UPDATE versus DELETE. All of the users and groups should be members of the HR_USERS group.

At the time of connection, the HR_USERS group is used to identify the correct shared login for each underlying database connection. If the connecting user is a consuming member of another shared login, the GROUP value would properly identify which shared login 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 in 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 selection of outbound credentials. The basic algorithm selects a map based on the proximity of the specified group to the map.

Candidate Map Processing

Candidate maps are processed based on one of the following criteria:

- If the user is not a direct or indirect member of the shared login consumer group, the map is not a candidate.

- If the GROUP is not a direct or indirect consumer of the map, the map is not a candidate.

- The distance from the GROUP to the map is determined by following the group member-of relationship all the way to the group that is the direct consumer. The candidate map is retained 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, use one of the following resolutions:

- If exactly one map has been retained, use the associated credentials.

- If two or more maps were retained, check the closest two and use the credentials associated with the closest of the two (error if the distances are the same unresolved conflict).

- If no maps have been retained, use no 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 so that users qualify for a particular shared login, for example:

- The administrator wants to grant access to the Oracle account EXECUTIVE_USER to his most privileged users, identified by the MARKETING_EXECUTIVE group.

- The administrator wants to grant access to the 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 the Oracle account STANDARD_USER to all other known users in the system, identified by the SASUSERS group.

- The administrator created groups that 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 SASUSERS consuming groups, respectively, to these shared logins.

Then, the shared login chosen is 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 SASUSERS 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 SASUSERS 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 SASUSERS 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

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 in 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 selection of outbound credentials. The basic algorithm selects a map based on the proximity of the specified group to the map.

Candidate Map Processing for a User

One of the following resolutions determines candidate map processing:

- If the user is not a direct or indirect consumer of the map, the map is not a candidate.

- 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 the distance is less than or equal to the current minimum distance to the map. The current minimum distance is updated.

Use one of the following resolutions after all candidate maps have been processed:

- If exactly one has been retained, use the associated credentials.

- If two or more were retained, check the closest two and use the credentials associated with the closest of the two (error if the distances are the same unresolved conflict).

- If maps have not been retained, use no credentials.

Candidate Map Processing for SASUSERS
- If SASUSERS 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 to Candidate Map Processing for PUBLIC.

Candidate Map Processing for PUBLIC
- If SASUSERS 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 were retained.

Path Length Computation Details
If SASUSERS or PUBLIC is a member of another group, and that group is a map consumer, direct or indirect, the path length does not increment 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 which is a quick way to place all users in that group.

Working with Catalogs and Schemas

Working with Catalogs

About Catalogs
Databases retain a structure that contains data stored in a database. 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**.

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 expose data. For other data services, catalogs and schemas are defined in the data source, and catalog and schema names can be registered in SAS Federation Server to reflect those objects.

Registering Catalogs
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.

Catalog Name Mapping

If your database supports native catalogs, you can use catalog mapping to avoid duplication errors. Certain SAS Federation Server drivers, such as Netezza and ODBC, provide a connection option, CATALOG= that facilitates catalog name mapping. Using CATALOG=, you can specify an arbitrary identifier for an SQL catalog that groups logically related schemas. For databases that do not support native catalogs, any identifier is valid, for example, `catalog=myodbc`. For databases such as SQL Server that do support native catalogs, 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 database. For example, to map the logical catalog `mycat` to the native catalog named `newusers`, use the following command: `catalog=(mycat=newusers);`. If a catalog name is not specified, then the native catalog name is surfaced to the users. Catalog name maps can be used only with FedSQL. They are not valid with native SQL.

Working with Schemas

About Schemas

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. 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. A schema is approximately equivalent to a SAS library.

Creating and Registering a Schema

You can create a schema for the Base SAS, MDS, and SASHDAT data sources. The following is an example of the CREATE SCHEMA DDL statement:

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

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 permissions to a user or group on a schema.

A complete list of options is shown in CREATE SCHEMA DDL on page 258.

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. As a schema owner, certain privileges are automatically granted to the schema owner.

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**.

- The schema owner can alter the schema's configuration options.

- Only the schema owner can change a view from invoker to definer and vice versa.

- The schema owner can publish and drop DS2 packages. However, this restriction can be lifted, granting publish/drop rights to all users by setting PL Source Management Security on the server. See the "Administration DDL", PL (Procedure Language) Source Management Security server option.

- When schema ownership changes, the previous owner receives default privileges from the schema's container, whether it is a catalog or a data service. However, explicitly denied privileges remain in tact for the schema.

- Schemas should not be owned by a system user.

Additionally, an administrator 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. GRANT to a 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.

Chapter 8

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 account is available and shared logins are configured and ready for use. For additional information, see "SAS Metadata Server".

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

Federated SQL Views

Overview

When there is a need to view information from multiple data sources or other source types, you can create a reusable federated SQL view (FedSQL view) to deliver data from multiple relational and non-relational sources. A federated SQL 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 a federated SQL view, the FedSQL dialect must be selected in the DSN.

Federated SQL 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 federated SQL 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 federated SQL views for SAS Federation Server:

• **Invoker's rights view**: An invoker's rights view is run with the invoking user's credentials.

• **Definer's rights view**: A definer's rights view is run with the credentials of the schema owner.

The invoker's rights 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.

A definer's rights view allows security management 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 a definer's rights view, a view is created as an invoker's rights view by default.

Here is an example of creating an invoker's rights 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 for which he has the SELECT privilege only. You can create a definer's rights view using the following example syntax:

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

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

```
ALTER VIEW
          view1 SECURITY INVOKER
```

```
ALTER VIEW
          view1 SECURITY DEFINER
```

Requirements for Definer's Rights Views

Here are the requirements for definer's rights views:

- Use of definer's rights views requires a trust relationship between SAS Federation Server and SAS Metadata Server. Trust is established when the connection uses a Trusted User Account. This enables impersonation of schema owners during execution of SQL. Specifically, the capability is required to retrieve:

  1. group memberships required for SQL authorization enforcement of data accessed in execution of a definer's rights view, and

  2. outbound database credentials forwarded to make transient connections used during view execution.

  After defining a Trusted User Account, specify the Trusted User in the Federation Server by setting the trusted user account and password using 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 using 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**, or **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 View Schema Owner (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 Federated SQL 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 "Working with Schemas" on page 123.

Creating Federated SQL 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. To specify a definer's rights view, replace INVOKER with DEFINER:

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

Create a Federated SQL 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 federated SQL view that allows access to data sources. With dynamic connections you can also connect to a data source to find DS2 functions, or execute the Data Quality Methods. 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 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 federated SQL views for 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 the required logins to the data sources that require dynamic connections. 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 be a personal or shared login.

Secured Connections

Connections made dynamically during view execution are secured. For example, 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.
```

A SELECT privilege check is not 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 FedSQL definer'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. When 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 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, returning 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.

When a definer's rights view is executed, it uses the credentials of the view's schema owner, rather than the current user's credentials, 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. This allows a single copy of the view result set to be cached and consumed by all users. You can use SAS Federation Server authorization,

including table-, column-, and row-level security, 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 cache for the view is dropped.

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: The following table 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.
- **VS_Owner** (View Schema Owner) is the owner of the schema where the cached view resides.
- **CS_Owner** (Cache Schema Owner) is the owner of the schema where the cache tables reside.

Table 8.1 *Requirements for Cached Views*

Who	Action	Privilege on Object
User	Create a cache Create or drop cached views within a schema Refresh cached views within the schema	CONNECT on ADMIN DSN CREATE CACHE on the VS ALTER CACHE on the VS
CS_Owner – owns all cache tables in its schema, and this identity is used for executing the view and saving the cached data.[1]	Assigned to CS object Execute, Save cached data	CONNECT on Data service that contains cache tables.
VS_Owner – owner of the schema where the cached view resides.	Cache results sets in the CS for views owned by the VS_Owner.[2]	CREATE TABLESPACE on CS

[1] The CS_Owner 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 CS_Owner user during cache creation and refresh, and the CS_Owner must be able to select from the original view. During data cache connections, the CS_Owner connects to the databases that contain the CS and the VS using a credential search order (CSO) of "PERSONAL, SHARED".

[2] The server assumes the identity of the CS_Owner 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.

Note: Administrators implicitly have all privileges, including CREATE CACHE and ALTER CACHE.

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 *SAS Federation Manager: User's Guide*.

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

Creating a Cache
`CREATE CACHE "catalog"."schema"."view" IN`
`"cache-catalog"."cache-schema"`
Privileges: CREATE CACHE
Information Views: CACHES, OBJECTS, CONFIG_OBJECTS

> Use the CREATE CACHE DDL statement to cache a definer's rights view or change an existing cache definition. 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.

> Several options are available with the CREATE CACHE statement. For example, you can specify that a cache refreshes at server start up by specifying the **SET RESTART='REFRESH'** when creating the cache.

Alter a Cache
`ALTER CACHE "catalog"."schema"."view" REFRESH | DISABLE |`
`ENABLE`
Privileges: CREATE CACHE, ALTER CACHE (On the view)
Information Views: CACHES

> To alter an existing cache, use the ALTER CACHE DDL statement. 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 when the refresh is done. Use DISABLE and ENABLE to temporarily disable caches.

Disabling and Enabling Caches
A cached view can be temporarily disabled if it has to be taken offline. 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. When the cached view is enabled, users are transparently directed back to the actual cached view.

`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. During the time that 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.

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

When 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.

Purge Cache

PURGE CACHE

Privileges: System user, Administrator, CREATE CACHE (on the server object)

Purge Cache forces the removal of cache tables that are no longer in use. Only system users, administrators, or those with CREATE CACHE privilege on the server object can execute this DDL statement. There are two commands that you can use to purge cache tables:

- Use the explicit PURGE CACHE DDL statement 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.

- 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. The default is 30 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 wakes 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 cache views might be in use at the time of cleanup.

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

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

Drop Cache

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

Privileges: CREATE CACHE (on the view)

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

Understanding Data Federation and Best Practices

Overview

Successful data federation projects require careful preparation and attention in two areas:

- data model

- data security

Proper understanding of your organization's data access needs is key to a successful data federation deployment. The data model controls 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 incorrect results if the data is not well understood. Data security design should consider how your users access the back-end data. Will users access the data under their individual authorizations? Will you establish one or more data owners who 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. However, 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 processed in real time during the data requests. These operations should be as efficient as possible and require a well-planned 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, such as 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, although 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 should 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 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 so that users can access data with personal credentials can be burdensome. The security capabilities of SAS Federation Server provide some alternatives that can help ease security administration.

Invoker's Rights Views

If 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 require the invoking user to have proper credentials to access the data. In addition, you should 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 needs to ensure that proper privileges are assigned to each invoker of the view on the data source, as well as the FedSQL view itself. The privileges are granted to a group, or set of groups for which the invokers are members, instead of to each individual invoker.

Because users must be able to directly access all of their data, you should secure each object for each user. Also, 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, as shown in the following 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, User1 receives an error when attempting to access table T. In this case, you might consider denying SELECT privilege on the view to User1. Furthermore, if User2 has SELECT privilege on table T1 and column C1 but not on column C2, then User2 receives an error when selecting from the view. Assigning column-level security on the view that denies SELECT privilege on column C2 to User2 does not help, because the underlying view definition specifically requires access to column C2 in table T. When creating views that require data from other objects with column-level security, you might consider selecting all columns using an asterisk (`*`). Here is an example:

```
SELECT * FROM T
```

When the Federation Server is configured with **SelectStarExpansion=VISIBLE**, the `*` expands all columns for which the invoking user has SELECT privilege. This enables you to create a single view that can be used by all invokers, yet each invoker sees only the columns for which the invoker has SELECT privilege.

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 SAS Federation Server to access the data under the rights of the defining user. This is accomplished by creating definer's rights FedSQL views. When these views are executed by a user, connections to back-end data are made under the definer of the view. The definer is actually the owner of the schema that contains the view.

In this security model, you typically deny access to the back-end data to all users except the view definer. Any column-level, table-level, 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 should 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 security settings can then be consolidated on the exposed objects of the data model, which are usually FedSQL views. Then users access the views only, and 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 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 might also affect indexes on the table, consider issuing a `DELETE FROM TABLE` command to delete all the rows from the table. In doing this, the underlying table remains intact, as well as 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, data in back-end data sources is being manipulated (created and dropped) by a few power users only. These users 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. However, if designed correctly, the DROP privilege can be used from the container object.

Business users are typically the ones requiring table-level 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-level, column-level, and row-level security on back-end data source objects.

Chapter 9
Data Quality

Overview

Data Quality on SAS Federation Server is implemented though SAS Quality Knowledge Base (QKB) using FedSQL and DS2. The data quality methods use data quality rules from the SAS QKB in order to cleanse data. The rules, referred to as QKB definitions, are operation- and locale-specific. The FedSQL driver is required to process data quality functions on SAS Federation Server. By default, the data quality functions are exposed through an MDS (Memory Data Store) table:

- Catalog: SYSPROC
- Schema: SYSPROC.DQ

About QKB

The SAS Quality Knowledge Base (QKB) is a collection of files that store data and logic that define data management operations such as parsing, standardization, and matching. SAS software products refer to the QKB when performing data management operations, also referred to as data cleansing, on your data. Each SAS QKB is defined by a locale that specifies the language or character set that is used for managing different types of data. The examples in this chapter are based on the English, USA (ENUSA) locale.

There are several types of definitions in SAS QKB. The definition types available in SAS QKB that are exposed in DS2 are as follows:

- **Case Definitions**: Use case definitions to apply uppercase and lowercase lettering using context-sensitive rules.

- **Extraction Definitions**: Extraction definitions are used to extract specific entities or attributes from a text string.

- **Gender Definitions**: Use gender definitions to determine the gender of a person from his or her name or other information.

- **Identification Definitions**: Identification definitions determine the type of data that is represented by a text string.

- **Match Definitions**: Use match definitions to generate a matchcode for a text string.

- **Parse Definitions**: Use parse definitions to segment a string into several parts.

- **Pattern Definitions**: Use pattern definitions to return a simple representation of a character pattern based on a text string.

- **Standardization Definitions**: Standardization definitions generate a preferred standard representation of a string, presenting a consistent format for data.

For complete details, see the Help that is delivered with SAS QKB.

About the Data Quality Methods

Overview

Data quality methods implement the basic QKB definition types. Each data quality method is defined in **dfs_serv_dq.xml** with an associated QKB path and locale. The configuration file contains each of the data quality methods presented below. However, you can create custom wrappers for definitions that are not presented in this topic. All of the data quality methods have the same general syntax, except DQPARSE, DQEXTRACT, and DQMATCH. DQPARSE and DQEXTRACT require an additional input token that qualifies the output field. DQMATCH uses a sensitivity code that specifies the degree of similarity for the data matching.

Standardization

DQSTANDARDIZE

You can perform standardization using the **DQSTANDARDIZE** method:

```
method dqstandardize(nnvarchar(256) value, nnvarchar(256) qkb_def,
nnvarchar(50) locale) returns nnvarchar(256);
```

The **DQSTANDARDIZE** method supports the following data types for the value parameter:

```
nnvarchar(256)|date|timestamp
```

Standardization generates a preferred standard representation of data values. Standardization definitions are provided for character content such as dates, names, and postal codes. The available standardization definitions vary from one locale to the next. The return values are provided in the appropriate case, and insignificant blank spaces and punctuation are removed. The order of the elements in the return values might differ from the order of the elements in the input character values.

Here are sample SELECT statements for standardization:

```
SELECT SYSPROC.DQ.DQUALITY.DQSTANDARDIZE (
          STATE,
          'State/Province (Full Name)',
          'ENUSA' )  AS STANDARD_STATE
FROM employee

SELECT SYSPROC.DQ.DQUALITY.DQSTANDARDIZE (
          postalCode,
          'Postal Code',
          'ENUSA' )  AS STANDARD_POSTAL_CODE
FROM employee
```

Matching

DQMATCH

You can perform matching using the **DQMATCH** method:

```
method dqmatch(nvarchar(256) value, nvarchar(256) qkb_def, int sensitivity,
nvarchar(50) locale) returns nvarchar(256);
```

The **DQMATCH** method supports the following data types for the value parameter:

```
nvarchar(256)|date|timestamp
```

Matching analyzes the input data and generates a matchcode for the data. The matchcode represents a condensed version of the character value. Similar strings receive identical matchcodes. You can specify a sensitivity value, ranging from 0–100, indicating the degree of similarity that should be applied to consider something a match. A sensitivity value of 100 yields more information, and 0 yields less. The default recommended sensitivity value is 85. Here are sample SELECT statements for matching:

```
SELECT SYSPROC.DQ.DQUALITY.DQMATCH (
          postalCode,
          'Postal Code', 85,
          'ENUSA' )  AS MATCH_POSTAL_CODE
FROM employee


SELECT SYSPROC.DQ.DQUALITY.DQMATCH (
          phone,
          'Phone', 50,
          'ENUSA' )  AS MATCH_PHONE
```

```
FROM employee
```

Pattern Analysis

DQPATTERN

You can perform pattern analysis using the **DQPATTERN** method:

```
method dqpattern(nvarchar(256) value, nvarchar(256) qkb_def,
nvarchar(50) locale) returns nvarchar(256);
```

Pattern analysis returns a simple representation of a text string's character pattern, which can be used for pattern frequency analysis in profiling jobs. Pattern analysis identifies words or characters in the input data column as numeric, alphabetic, non-alphanumeric, or mixed. The choice of pattern analysis definition determines the nature of the analysis. Here are sample SELECT statements for pattern analysis:

```
SELECT SYSYPROC.DQ.DQUALITY.DQPATTERN (
            name,
            'Word',
            'ENUSA' )  AS PATTERN_WORD
FROM employee


SELECT SYSPROC.DQ.DQUALITY.DQPATTERN (
            address,
            'City - State/Province - Postal Code',
            'ENUSA' )  AS PATTERN_CITY_STATE_POSTAL
FROM employee
```

Identification Analysis

DQIDENTIFY

You can perform identification analysis using the **DQIDENTIFY** method:

```
method dqidentify(nvarchar(256) value, nvarchar(256) qkb_def,
nvarchar(50) locale) returns nvarchar(256);
```

Identification analysis returns a value that indicates the category of the content in an input character string. The available categories and return values depend on your choice of identification definition and locale. Here are sample SELECT statements for identification analysis:

```
SELECT SYSPROC.DQ.DQUALITY.DQIDENTIFY (
            Name,
            'Field Name',
            'ENUSA' )  AS IDENTIFY_FIELD_NAME
FROM employee


SELECT SYSPROC.DQ.DQUALITY.DQIDENTIFY (
            email,
            'E-mail (Country Identification)',
            'ENUSA' )  AS IDENTIFY_EMAIL
FROM employee
```

Gender Analysis

DQGENDER

You can perform gender analysis using the **DQGENDER** method:

```
method dqgender(nvarchar(256) value, nvarchar(256) qkb_def,
nvarchar(50) locale) returns nvarchar(256);
```

Gender analysis evaluates the name or other information about an individual to determine the gender of that individual. If the evaluation finds substantial clues that indicate gender, the function returns a value that indicates that the gender is female or male. If the evaluation is inconclusive, the stored procedure returns a value that indicates that the gender is unknown. The exact return value is determined by the specified gender analysis definition and locale. Here is a sample SELECT statement for gender analysis:

```
SELECT SYSPROC.DQ.DQUALITY.DQGENDER (
          NAME,
          'Name',
          'ENUSA' )  AS GENDER_NAME
FROM employee
```

Casing

DQLOWERCASE

The **DQLOWERCASE** method applies lowercase text:

```
method dqlowercase(nvarchar(256) value, nvarchar(256) qkb_def,
nvarchar(50) locale) returns nvarchar(256);
```

Casing applies context-sensitive case rules to text. It operates on character content, such as names, organizations, and addresses. Here is a sample of lower casing:

```
SELECT SYSPROC.DQ.DQUALITY.DQLOWERCASE (
          name,
          'Lower',
          'ENUSA' )  AS LOWERCASE_PHONE
FROM employee
```

DQUPPERCASE

The **DQUPPERCASE** method applies uppercase text:

```
method dquppercase(nvarchar(256) value, nvarchar(256) qkb_def,
nvarchar(50) locale) returns nvarchar(256);
```

Here is a sample of upper casing:

```
SELECT SYSPROC.DQ.DQUALITY.DQUPPERCASE (
          name,
          'Upper',
          'ENUSA' )  AS UPPERCASE_PHONE
FROM employee
```

DQPROPERCASE

The **DQPROPERCASE** method applies uppercase and lowercase text using context-sensitive rules:

```
method dqpropercase(nvarchar(256) value, nvarchar(256) qkb_def,
nvarchar(50) locale) returns nvarchar(256);
```

Parsing

DQPARSE

You can perform parsing using the **DQPARSE** method:

```
method dqparse(nvarchar(256) value, nvarchar(256) qkb_def,
 nvarchar(256) tokener, nvarchar(50) locale) returns nvarchar(256);
```

The **DQPARSE** method supports the following data types for the value parameter:

```
nvarchar(256)|date|timestamp
```

Parsing segments a string into semantically atomic tokens. Parsing is performed with the **DQPARSE** method. Here are samples of the SELECT statement for parsing:

```
SELECT SYSPROC.DQ.DQUALITY.DQPARSE (
          address,
          'Address', 'Street Name',
          'ENUSA' )  AS PARSE_ADDRESS_STREET_NAME
FROM employee


SELECT SYSPROC.DQ.DQUALITY.DQPARSE (
          name,
          'Name (Global)', 'Prefix',
          'ENUSA' )  AS PARSE_NAME_PREFIX
FROM employee
```

Extraction

DQEXTRACT

You can perform extraction using the **DQEXTRACT** method:

```
method dqextract(nvarchar(256) value, nvarchar(256) qkb_def,
 nvarchar(256) tokener, nvarchar(50) locale) returns nvarchar(256);
```

Extraction returns one or more extracted text values, or tokens, as output.

Data Types

The table below describes the parameters associated with the data quality methods.

Parameter	Data Type	Description
qkb_def	nvarchar(256)	Quality knowledge base definition as defined in SAS QKB.
value	nvarchar(256)	Primary input string for modification with a data quality method. Data quality methods silently truncates character strings that are larger than 256.

Parameter	Data Type	Description
sensitivity	integer	Parameter that controls the sensitivity of the match function. Varies from 0-100.
tokenlist	dqtokens	Data structure used to store multiple strings resulting from using the **dq.parse** and **dq.extract** data quality method. The other data quality functions are known as scalar functions.

Executing the Data Quality Methods

The data quality methods referenced above are stored in the **dfs_serv_dq.xml** configuration file. The code in the configuration file is stored in an MDS package 'SYSPROC.DQ.DQUALITY' at server start-up. Upon execution, the code is read from the MDS package and executed with the provided parameters. The EXECUTE privilege is granted to the SASUSERS group for the DQ schema. The following example uses **DQSTANDARDIZE** to outline the steps for working with the data quality methods.

1. Implement a data quality method by extracting the code from **dfs_serv_dq.xml**.

```
method dqstandardize(nvarchar(256) value, nvarchar(256) qkb_def, nvarchar(50) locale)
 returns nvarchar(256);

/* Set the QKB path. */
        dq.setQKB('&cfg.qkb_loc;');
        if (check_err()) then return null;

        /* Load locale. */
        dq.loadLocale(locale);
        if (check_err()) then return null;

        /* cleanse data */
        value = dq.standardize(qkb_def, value );
        if (check_err()) then return null;

        return value;

    end;
```

2. Use a FedSQL procedure to call the method that was implemented in step 1:

```
proc fedsql noprompt=&connectionString nolibs noerrorstop;
    create table schema.cleansedTable as SELECT
    SYSPROC.DQ.DQUALITY.DQSTANDARDIZE (
            name ,
            'Name',
            'ENUSA' )  AS Standardized

    FROM schema.inputTable;
    quit;
```

The data quality methods on SAS Federation Server are designed to run with FedSQL. When you run a data quality method from a SELECT statement in a FedSQL view, you are dynamically connected to the SYSPROC catalog if the catalog is not present in your DSN connection. However, if you issue a SELECT statement outside of a FedSQL view, you are not dynamically connected to the SYSPROC catalog. To avoid an error, you must create a federated DSN and include the SYSPROC catalog.

Customizing QKB

The standard definitions in the SAS Quality Knowledge Base are sufficient for performing most data quality operations. However, you can use DataFlux Data Management Studio to customize the QKB by modifying definitions or creating new definitions for use with your own business data. For more information about customizing QKBs, see the "Managing Quality Knowledge Bases" chapter in the *DataFlux Data Management Studio: User's Guide*.

If you want to customize your QKB, then as a best practice, you should customize your QKB on a local workstation before copying it to the server for deployment. When updates to the QKB are required, you can merge your customized content into the updated QKB locally and deploy a copy of the updated, customized QKB to SAS Federation Server. You must copy the customized QKB to the directory on SAS Federation Server that contains the QKB. This is usually the path reflected in the **dfs_entities.dtd** file in the **cfg.qkb_loc ENTITY**. The QKB path is enclosed in quotation marks. Here is an example:

```
<!-- Data Quality functions configuration -->
<!ENTITY cfg.qkb_loc   "\\tstsrc\tst\dev\tst-v940m3\tktest\testmisc\QKB\CI24">
```

You must restart SAS Federation Server to load the new QKB.

See the online Help provided with your SAS Quality Knowledge Base for information about how to merge any customizations that you have made into an updated QKB.

QKB Documentation

The online documentation for a Quality Knowledge Base is installed with the software. You can use a web browser to open the documentation at one of the following default locations.

For QKB CI 22 and later:

```
drive:\Program Files\SAS\QKB\<type><version>_<unique_identifer>\doc\html\qltykb1000.html
```

Chapter 10

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. Requests can be in the form of FedSQL statements or the SQL statements that are the implementation of the data service.

The supported data sources and connection options are presented in the following topics. For information about data type support, see the *SAS FedSQL Reference Guide*.

SAS Federation Server Driver for Apache Hive

About the SAS Federation Server Driver for Apache Hive

The SAS Federation Server Driver for Apache Hive (Driver for Hive) allows SAS Federation Server to query and manage large data sets that reside in distributed storage. As with other drivers, to realize the full benefits of the SAS Federation Server such as security and data federation, you should use the FedSQL driver (dialect) with the driver. Using FedSQL, the Driver for Hive supports Read, and Insert using Bulk Operations, but the Driver for Hive does not support Write operations such as update, delete and index creation. For additional information about FedSQL, see the *SAS FedSQL Language Reference*.

The driver also supports a native dialect (HiveQL). Check your Hive version to determine functionality. For example, if using a version of Hive that supports the ORC file format, you should be able to Insert, Delete, and Update.

You can use Kerberos with the Driver for Hive by specifying the authentication mode option in a connection string. For more information, see AUTH_MODE in the connection options below.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source.

Hadoop JAR files must be installed and the SAS_HADOOP_JAR_PATH environment variable defined before using the Driver for Hive. The variable points to the location of the Hadoop JAR files, and is defined using the **SetEnv** option set in the **dfs_serv.xml** configuration file. Here is an example:

```
<OptionSet name="SetEnv">
     <Option name="SAS_HADOOP_JAR_PATH">\SAS\Config\Lev1\FederationServer
      \lib\Hadoop</Option>
 </OptionSet>
```

If the JAR file location changes, you must update the SAS Federation Server configuration file with the new location.

Note: The SAS Deployment Wizard installs the necessary Hadoop JAR files and sets the SAS_HADOOP_JAR_PATH environment variable when the Driver for Hive is included with a SAS Federation Server plan.

Data Service Connection Options for Hive

Overview

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service, which provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Connection Options

You can specify one or more connection options when defining a DSN or data service. The Driver for Hive supports the following connection options.

Option	Description
DRIVER	**DRIVER=HIVE** Required. Identifies the HIVE data source to which you want to connect.
SERVER	**SERVER='server-name'** Specifies the host name of the Hive server. If the server name contains spaces or nonalphanumeric characters, enclose it in quotation marks.
PORT	**PORT=port_number** Specifies the port number that is used to connect to the specified Hive Server. The default is 10000.
SUBPROTOCOL	**SUBPROTOCOL=Hive|Hive2** Specifies whether you are connecting to a Hive service or a HiveServer2 (Hive2) service. The default is Hive2.
CATALOG	**CATALOG=catalog-name** Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas.
SCHEMA	**SCHEMA=hive-schema-name** Specifies a Hive schema name that is used to define a name other than 'default'. Alias: DATABASE, DB
AUTH_MODE	**AUTH_MODE=default | Kerberos** Specifies the authentication mode for the connection. The options are default and Kerberos. If using Kerberos for the authentication mode, you must specify the Hive principal host name using the HIVE_PRINCIPAL connection option.

Option	Description
HIVE_PRINCIPAL	**HIVE_PRINCIPAL=service-principal-hostname** Specifies the Hive principal string in an environment that uses Kerberos(for example, **HIVE_PRINCIPAL=hive/_HOST@UNX.SAS.COM**). Required with AUTH_MODE=Kerberos.
USER_PRINCIPAL	**USER_PRINCIPAL=** Specifies that the HDFS path and JDBC path use JAAS to perform a doAs for the given Kerberos user principal. Alias: **auth_mode=Kerberos;uid=**
UID	**UID=(user-name)** Specifies the user name with the necessary permissions to perform Read and Write operations. UID and PWD are not needed when connecting to a server that uses Kerberos authentication. Alias: USER
PWD	**PWD='user-password'** Specifies a password that correlates with the user ID (UID) value. If the password contains spaces or nonalphanumeric characters, enclose it in quotation marks. UID and PWD are not needed when connecting to a server that uses Kerberos authentication. Alias: PASSWORD

PROPERTIES (JDBC session configuration properties)	Use the PROPERTIES option to specify one or more JDBC connection properties to override the default JDBC connection properties. In a JDBC URL, custom properties are separated from the default properties by the question mark (?) character. Multiple properties are separated by the semicolon (;) character. Here is an example:
	
	`PROPERTIES="hive.default.fileformat=ORC;hive.exec.compress.intermediate=true"`
	
	Site-wide Hive properties are specified in the **hive-site.xml** file in the Hive configuration directory.
	
	You can use the properties option to set the default file format for managed and unmanaged tables respectively. The following example specifies optimized row columnar (ORC) as the default file format when creating a table:
	
	`PROPERTIES=hive.default.fileformat.managed=orc`
	`PROPERTIES=hive.default.fileformat=orc`
	
	Note: The ORC file format is available beginning with Hive version 0.11.
	
	The following example sets the partition mode to nonstrict, which allows dynamic inserts against a partitioned table (for example, when a static partition has not been explicitly defined in the SQL statement).
	
	`PROPERTIES=hive.exec.dynamic.partition.mode=nonstrict`
	
	Note: These Hive–defined properties can be changed or removed by Hadoop vendors at any time.

Option	Description
HD_CONFIG	**HD_CONFIG=path to hadoop configuration file** Specifies the name and path for the Hadoop cluster configuration file. This file contains entries for Hadoop system information, including file system properties such as fs.defaultFS. The configuration file can be a copy of the Hadoop core-site.xml file. However, if your Hadoop cluster is running with HDFS failover enabled, you must create a file that combines the contents of the Hadoop core-site.xml and hdfs-site.xml files. *Note:* The name and path values for the Hadoop cluster configuration file are normally set in the SAS_HADOOP_CONFIG_PATH option in the dfs_serv.xml configuration file. This is the recommended configuration method. Use HD_CONFIG only if you want to override the settings in the server configuration file. See "SAS Federation Server Configuration Reference" for details.
HDFS_TEMPDIR	**HDFS_TEMPDIR='path'** Specifies the path to the HDFS directory that is used for read and write of temporary data. The default is **HDFS_TEMPDIR='/tmp'**
DBMAX_TEXT	**DBMAX_TEXT=32767** Specifies the length for a string data type. The maximum length is 2 gigabytes. The default is 32767.
LOGIN_TIMEOUT	**LOGIN_TIMEOUT=number_of_seconds** Specifies a login time-out, in seconds, for non-responsive connections. A value of 0 indicates that there is no time-out and the connection will 'wait forever'. The default value is 30 seconds.
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 as follows: • **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 is 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.

ODBC Apache Hive Wire Protocol Driver Usage Notes

Configuring ODBC Options

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.

You cannot use the unixODBC driver manager if the wire protocol drivers are installed.

Note: The behavior of a DSN using a wire protocol driver with the catalog option selected, returns only the schemas that have associated tables or views. To list all existing schemas, create a DSN without the catalog option selected.

When configuring an ODBC DSN using the Apache Hive Wire Protocol driver, select the following options on the **Advanced** tab:

• **Remove Column Qualifiers**

   Note: This option might be appended with **(Microsoft Access Compatibility)** in the ODBC Administrator.

Setting the Maximum Character String Size for Hive

Hive has a single data type for storing text, **STRING**, which is a variable-length character string with a maximum size of 2G. As a result, this can create very large character fields when processing data. Since Hive's string type is comparable to **VARCHAR** in other data sources, you can set the ODBC attribute, **Max Varchar Size** to specify the maximum character string size. Set the **Max Varchar Size** value using Advanced Options in Windows ODBC Administrator, or in UNIX by editing **odbc.ini** in the specified path or $HOME directory.

You can also specify this option in a connection string using the **CONOPTS** container. Here is an example:
DRIVER=ODBC;DB=hive;UID=dbitest;PWD=dbigrp1;SCHEMA=default;CON OPTS=(MaxVarcharSize=300);CATALOG=FOO;.

SAS Federation Server Driver for Base SAS

About the SAS Federation Server Driver for Base SAS

The SAS Federation Server Driver for Base SAS (Driver for Base SAS) is a SASProprietary driver that provides Read and Update access to legacy SAS data sets. With the SAS Federation Server Driver for Base SAS you can create SAS data sets that can be accessed by both the legacy and SAS Federation Server data access services.

The Driver for Base SAS supports much of the Base SAS functionality, including SAS indexing and general integrity constraints, as well as much of the Federated Query Language (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.

The SAS Data Set

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**.

Metadata Bound Libraries

The Driver for Base SAS supports metadata–bound libraries and data sets. Additional connection options are not required to access the data. Since user access is controlled through permissions granted on SAS Metadata Server, it is recommended that users are granted all permissions on the metadata–bound library catalogs and schemas that reside on SAS Federation Server.

Data Service Connection Options for SAS Data Sets

Connection Options

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

To connect to a SAS data set, you must specify a schema, catalog, and a primary path in your DSN or connection string. These options are described in the connection options below.

The following connection options are supported for SAS data sets:.

Option	Description
DRIVER	**DRIVER=BASE;**
	Specifies the BASE data service to establish connection to a SAS data set. DRIVER is a required option.
CATALOG	**CATALOG=catalog-name;**
	Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. A catalog name can be up to 32 characters long. You must specify a catalog.
	Note: SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
(SCHEMA) NAME	**NAME=schema-name;**
	Specifies an arbitrary identifier for an SQL schema. 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. You must specify a schema.
PRIMARY PATH	**PRIMARYPATH=physical-location;**
	Required. 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.
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 with assigned attributes.

Advanced Connection 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 optional 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. 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 data loss if there is a crash.
CT_PRESERVE	**CT_PRESERVE = STRICT	SAFE	FORCE	FORCE_COL_SIZE**
	Allows users to control how data types are mapped. Note that data type mapping is 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 as follows:
	• **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned.
	• **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, 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). NO 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. 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. 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;...)**
	
	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:
	
	• **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 is not 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 is 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). 0 is the default. 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.

PATH_BIND	**PATH_BIND=CONNECT⎮ACCESS** Specifies when and how schemas are validated during connection. CONNECT validates the entire connection string at the time of connection and returns an error if one or more schemas is invalid. ACCESS validates schemas when they are accessed so that processing continues regardless of errors in the schema portion of the connection string. ACCESS is the default for SAS Federation Server.
TIME_TYPE	**TIME_TYPE=YES⎮NO** Specifies the format used for date types. YES is the default behavior which returns the date type as DATES. When NO is specified, date types are formatted as DOUBLES.

SAS Federation Server Driver for DB2

About the SAS Federation Server Driver for DB2

The SAS Federation Server Driver for DB2 (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 Driver for DB2 also supports DB2 for z/OS® and DB2 for mid-range systems such as AS/400™, and System i® when using IBM DB2 Connect™ for data access.

The 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.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See Setting Environment Variables for additional information.

Data Service Connection Options for DB2

Overview
To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Note: When performing connections through DSNs or connection strings, SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.

Connection Options
You can specify one or more connection options when defining a data service and DSN. The Driver for DB2 supports the following connection options 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**). You must specify a catalog. For the DB2 database, this is a logical catalog name to use as an SQL catalog identifier. *Note:* SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.

Option	Description
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 Connection Options

SAS Federation Server Driver for DB2 supports the following advanced connection options 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 that is stored in DBCODEPAGE, differs 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. To prevent the client layer from converting data incorrectly, you must first determine the correct value for DB2CODEPAGE and then set the CLIENT_ENCODING= option to match the corresponding encoding value in DB2CODEPAGE.
	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 SAS Federation Server, 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 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.

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 is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.
DEFAULT_ATTR	**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: • **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.

<table>
<tr><td>0</td><td>Causes the driver to use client-side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one.</td></tr>
<tr><td>1</td><td>Causes the driver to always use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is not used.</td></tr>
<tr><td>2</td><td>(Default) Causes the driver to never use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is used if available. Otherwise, the cursor will be forward-only.</td></tr>
</table>

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). 0 is the default. 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 as follows: • **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 is 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.

DB2 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.

You cannot use the unixODBC driver manager if the wire protocol drivers are installed.

Note: The behavior of a DSN using a wire protocol driver with the catalog option selected, returns only the schemas that have associated tables or views. To list all existing schemas, create a DSN without the catalog option selected.

When configuring an ODBC DSN using the DB2 Wire Protocol driver, set the following advanced options:

• **Application Using Threads**

FedSQL Driver Reference

Overview

The FedSQL language driver supports the FedSQL dialect, as documented in the *SAS 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 includes an SQL processor which supports the FedSQL dialect. The main emphasis of the FedSQL driver is to support federation of data sources. 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 in SAS Federation Server. The FedSQL driver supports the FedSQL dialect over any data source. For example, if the SQL request is from a single data source that does not support a particular SQL function, the FedSQL processor guarantees implementation of the request.

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 is used on top of a native data source driver and supports various connection options. To specify these options, use the DEFAULT_ATTR option in the CREATE DSN statement. For example,

```
CREATE DSN MYDSN UNDER "Oracle Service" CONNECT
'DEFAULT_ATTR=(SQL_MAX_COL_SIZE=500);DRIVER=ORACLE'
```

Connection Options

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
DEFAULT_ATTR=(SQL_CURSORS=2)
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.

    0   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.

    1   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.

    2   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.

SQL_AC_BEHAVIOR=n
DEFAULT_ATTR=(SQL_AC_BEHAVIOR=0)
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.

    0   A value of **0** (default) means that no transactions are attempted under-the-covers and operations such as emulated UDPATE, DELETE or INSERT.

    1   A value of **1** means that FedSQL each SQL command is run as a single transaction to better support the correct behavior when AUTOCOMMIT is ON (where individual operations like UPDATE, DELETE and INSERT should be atomic).

    2   A value of **2** means that transactions are required. This option will fail if the underlying drivers do not support transactions.

SQL_MAX_COL_SIZE=n
DEFAULT_ATTR=(SQL_MAX_COL_SIZE=1048576)
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

SQL_STMT_MEM_LIMIT=n
DEFAULT_ATTR=(SQL_STMT_MEM_LIMIT=209715200)
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.

SQL_TXN_EXCEPTIONS=n
DEFAULT_ATTR=(SQL_TXN_EXCEPTIONS=2)
FedSQL connection handle option. Supports dynamic connections regardless of the specified transaction isolation. Possible values are 0 or 2 (default).

  0   Specify a value of **0** to disable support for dynamic connections.

  2   Specify a value of **2** to enable support for dynamic connections.

SQL_USE_EVP=n
DEFAULT_ATTR=(SQL_USE_EVP=0)
FedSQL statement handle option. This option optimizes the driver for large result sets. The possible values are 0 or 1. 1 is the default.

  0   Specify **0** to turn optimization OFF.

  1   Specify **1** to enable optimization (ON).

SQL_VDC_DISABLE=n
DEFAULT_ATTR=(SQL_VDC_DISABLE=1)
FedSQL statement handle option. This option is used to allow or disallow use of cached data for a statement. The possible values are 0 or 1. 0 is the default.

  0   Specify a value of **0** to enable cached data.

  1   Specify a value of **1** to disable cached data.

SQL_XCODE_WARN=n
DEFAULT_ATTR=(SQL_XCODE_WARN=1)
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, 1 or 2. The default is 0.

  0   Specify **0** to return an error if data cannot be transcoded.

  1   Specify **1** to return a warning if data cannot be transcoded.

  2   Specify **2** to ignore transcoding errors.

DEFAULT_CATALOG=catalog-name
Specifies the name of the catalog that is set as the current catalog when connecting to the data source. This option is useful for SQL Server connections and federated connections.

Federation Server (FEDSVR) Driver Reference

About the Federation Server Driver

The Federation Server driver (FEDSVR) enables you to define a connection from one SAS Federation Server to another SAS Federation Server. This connectivity can be useful for distributing workload or to federate data between various federation servers. To use the driver, you must first create a data service. The data service definition can include a remote DSN, which restricts access on the remote federation server to that of the DSN. If you do not include a DSN in the data service definition, the data service connects to all of the data services available on the remote federation server. In this scenario, you can create data service DSNs to access specific remote DSNs on the remote federation server. For additional information, see the "CREATE DATA SERVICE" in Appendix 1.

Note: When working with FedSQL views, note that views cannot be created, modified, and subsequently cached, if you are using a remote SAS Federation Server that is connecting with a SAS Federation Server data service. You should create and cache these views from a local SAS Federation Server only. FedSQL views should also be administered on a local SAS Federation Server (for example, changing a view from definer's rights to invoker's rights).

Connection Options

The following connection options are supported for the Federation Server driver.

Option	Description
DRIVER	**DRIVER=FEDSVR** Required. Specifies the driver for connection to a SAS Federation Server from another SAS Federation Server.
PROTOCOL	**PROTOCOL=BRIDGE** Specifies the protocol for the connection. At this time, BRIDGE is the only option and is the default.
PORT	**PORT=port-number** Required. Specifies the port number of the SAS Federation Server that you are connecting to. There is no default port number associated with this option. Therefore, PORT must be specified.
CONOPTS	**CONOPTS=(connection-string)** Required. Specifies the connection string to be passed to the federation server (for example, **CONOPTS=(DSN=mydsn)**).

Option	Description
PROXYLIST	**PROXYLIST=http-proxy-string** Optional. Specifies the proxy information needed to connect with an HTTP proxy. When specifying the HTTP proxy, you must use the encoded characters, **%2F** in place of a forward slash (for example, **http:%2F%2Fsaveferris.com**).
URI	**URI=address** Optional. Specifies a proxy with a URI instead of using the PROXYLIST option. If using both the URI and PROXYLIST options, URI takes precedence and overrides PROXYLIST.
SERVICE	**SERVICE=service-name** Optional. Specifies a name from the services file as an alternative to port. The services file is located in **/etc/services** on UNIX, and **c:\windows\system32\drivers\etc\services** on Windows.
SERVER	**SERVER=host-name** Required. Specifies the name of the SAS Federation Server that you are connecting to.
UID	**UID=user-id** Required. Specifies the user ID that is used for connection to the SAS Federation Server.
PWD	**PWD=password** Required. Specifies the password associated with the user ID that is used for connection to the SAS Federation Server.
APPLICATION NAME	**APPLICATIONNAME=Fed-server-name** Specifies a symbolic name for the client connecting to the SAS Federation Server. Use APPLICATION NAME to associate a client with records in SQL logging. This option corresponds to an entry of **X{Client.AppName}** in the SQL Logging configuration file. *Note:* If APPLICATION NAME is not specified on the connection string, the driver should pass on the current setting from the App.Name option specified in the server's configuration file.

This example demonstrates how to create a data service for the FEDSVR driver:
CREATE DATA SERVICE remote_fed_server TYPE FEDSVR DOMAIN D76586 REGISTER () VALIDATE YES {OPTIONS ADD CONOPTS (PORT '1234', SERVER 'D76586', DRIVER 'FEDSVR', CONOPTS (DSN OracleShared))}

Here is an example connection string that uses some of the options presented in the table above:
DRIVER=FEDSVR;SERVER=FedServer1;PORT=1234;UID=user_name;PWD=password;CONOPTS=(DSN=mydsn).

SAS Federation Server Driver for Greenplum

About the SAS Federation Server Driver for Greenplum

The SAS Federation Server Driver for Greenplum (Driver for Greenplum) enables SAS Federation Server to read and update Greenplum tables. In addition, the driver creates Greenplum tables that can be accessed by both SAS Federation Server and Greenplum.

The 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.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See "SAS Federation Server Driver for Greenplum" for additional information.

Data Service Connection Options for Greenplum

Overview

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Connection Options

The Driver for Greenplum supports the following connection options.

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**). You must specify a catalog. For the Greenplum database, this is a logical catalog name to use as an SQL catalog identifier.
	Note: SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
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**;
	Specifies the Federation Server driver for the Greenplum database. You must specify a driver.

Option	Description
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 Connection Options

The Driver for Greenplum supports the following advanced connection options.

Option	Description
ALLOW UNQUOTED NAMES	**ALLOW_UNQUOTED_NAMES=NO|YES;** Specifies whether to enclose table and column names in 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 lowercase.
CLIENT ENCODING	**CLIENT_ENCODING=cei;** Specifies a client encoding value that overrides the default. The default is UTF8.
CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.

Option	Description
DEFAULT_ATTR	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: • **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. <table><tr><td>0</td><td>Causes the driver to use client-side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one.</td></tr><tr><td>1</td><td>Causes the driver to always use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is not used.</td></tr><tr><td>2</td><td>(Default) Causes the driver to never use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is used if available. Otherwise, the cursor will be forward-only.</td></tr></table> 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). 0 is the default. 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 as follows: • **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 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 is 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.
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.

Option	Description
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.

You cannot use the unixODBC driver manager if the wire protocol drivers are installed.

Note: The behavior of a DSN using a wire protocol driver with the catalog option selected, returns only the schemas that have associated tables or views. To list all existing schemas, create a DSN without the catalog option selected.

When configuring an ODBC DSN using the Greenplum Wire Protocol driver, select the following options on the **Advanced** tab, if they are not already selected by default:

- **Application Using Threads**
- **Enable SQLDescribeParam**
- **Fetch TSFS as Time**
- **Fetch TSWTZ as Timestamp**

SAS Federation Server Driver for MDS

About the Memory Data Store (MDS)

Memory Data Store, or MDS, is a transactional in-memory data store that can be used with SAS Federation Server. MDS must be used with FedSQL. 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 be in use. Users can be connected to the database, but they cannot have a table open in the schema. Also, you cannot drop a table if it is referenced by a prepared statement or has a pending transaction with uncommitted changes.

MDS supports optimistic concurrency providing a snapshot transaction, which means that a transaction sees a consistent version of the data when the transaction is started. When an MDS transaction starts, the state of the database is logically frozen at that point in time. The transaction sees the database consistently but changes made by other transactions are not visible until the transaction is committed or rolled back, and its state synchronized so that it sees the new state of the database.

To access data in an MDS table, you must first configure an MDS data service and DSN.

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 241.

- Use the **New Data Service** function in SAS Federation Server Manager.

You can create multiple data services if needed. See "SAS Federation Server Driver for MDS" on page 171 for a list of connection options for MDS.

Each MDS data service catalog contains a pre-defined, read-only schema, named SYSTEMINFO. The SYSTEMINFO schema contains an auto-generated MEMORY table. You will need to define at least one additional schema to use MDS. To define a schema use the CREATE SCHEMA DDL statement or the **New Schema** function in SAS Federation Server Manager. Schemas cannot be modified, renamed or dropped while there are active connections to the database.

Data Service Connection Options for MDS

MDS supports the following connection string options.

Option	Description
LOCALE	**LOCALE=SAS locale identifer** Specifies the locale for message text and character conversion, both 'to' and 'from'.
ENCODING	**ENCODING=encoding-value** Specifies character encoding for the MDS table. The default is the encoding used for the SAS session. If SAS is not used, the operating system encoding is used as the default.
DB | DATABASE	**DATABASE=database name** Specifies the in-memory database instance. **DATABASE** must be specified if **CONOPTS=** is not specified. The database defaults to the catalog name if a database name is not specified.

Option	Description
CATALOG	**CATALOG=catalog name**; Specifies the catalog name. **CATALOG** must be specified if **CONOPTS=** is not specified.
CONOPTS	**CONOPTS=connection string options** Specifies the connection string options for the driver to cache in memory. If a connection string is not specified, the default is in memory only.
COMMIT	**COMMIT=Y│N** Specifies if the in-memory changes are written to the **CONOPTS=** driver. **COMMIT** must be used with the **CONOPTS=** option.
BULKLOAD	**BULKLOAD=Y│N** Specifies if data is inserted immediately, which bypasses transactions. The **BULKLOAD** option is valid only when **CONOPTS=** is not specified.
NUMERICS	**NUMERICS=Y│N** Allows numeric data types or treats them as double precision. The default is Y (Yes).
RETAIN	**RETAIN=Y│N** Specifies if the in-memory database is dropped after the last client disconnects. The default is N (No).
CT_PRESERVE	**CT_PRESERVE = STRICT │ SAFE │ FORCE │ FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, 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. **IDCASE** is valid only when **CONOPTS=** is not specified.

Option	Description
DEFAULT_ATTR	**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: • **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. <table><tr><td>0</td><td>Causes the driver to use client-side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one.</td></tr><tr><td>1</td><td>Causes the driver to always use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is not used.</td></tr><tr><td>2</td><td>(Default) Causes the driver to never use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is used if available. Otherwise, the cursor will be forward-only.</td></tr></table> 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). 0 is the default. 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. This option is valid only when **CONOPTS=** is not specified.
SCHEMAS	**SCHEMAS=("schema1";"schema2";"schema3")** Specifies a list of schemas defined in the database. Identify schema names with double quotation marks and separate each name by a semicolon.
SCHEMA	**SCHEMA=(NAME=schema-name1);SCHEMA=(NAME=schema-name2);...** Defines one or more schemas in the database. The default is a single schema using the defined catalog name.

Option	Description
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 **0** which specifies that there is no limit to the amount of memory used. **MAXDBMEM=0**.

MDS Database Memory

Limiting Memory Size

To limit the memory size for an MDS 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.

The MEMORY Table

The MEMORY table, **SYSTEMINFO.MEMORY**, contains information about memory usage and is always available. The table does not actually reflect how much data is in the table. Instead, it shows how much memory is being used by MDS to store the table, along with **MEM_PEAK** and **MEM_LIMIT**, if specified. The first row contains statistics about the MDS database. Subsequent rows provide information about each of the tables in the MDS database.

The MEMORY table includes the following columns:

Table 10.1 *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.

FedSQL Views and Data Caching with MDS

You can create federated SQL views and cache data from an MDS table. Because the data is in-memory and does not persist, view definitions are removed when the MDS table is dropped, if a REFRESH has not been set on the cache.

- If the cached view does not reside in MDS, the cache remains intact but reflects a status of deferred or inactive and can be refreshed.

- If the cached view resides in MDS, the view and cache objects that are stored in MDS are removed from the system tables.

If REFRESH has been configured on a cache, the cache refreshes at server start up:

- An in-memory cache is deferred at server start up.

- A cache that is set to refresh at start up is refreshed, even if it is disabled. If the refresh is successful, a deferred cache becomes active.

- A cache that is disabled, remains disabled after a refresh.

SAS Federation Server Driver for Netezza

About the SAS Federation Server Driver for Netezza

The SAS Federation Server Driver for Netezza (Driver for Netezza) enables SAS Federation Server to read and update legacy Netezza tables. In addition, the driver

creates Netezza tables that can be accessed by both SAS Federation Server and Netezza. Multiple schemas are supported for Netezza 7.0.3 and later.

The Driver for Netezza supports most of the FedSQL functionality.

The Driver for Netezza is a remote driver, which means that it connects to a server process in order to access data. The process might run on the same machine as SAS Federation Server, or it might run on another machine in the network.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See Setting Environment Variables for additional information.

Data Service Connection Options for Netezza

Overview

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Connection Options

The Driver for Netezza supports the following connection options.

Option	Description
CATALOG	**CATALOG=catalog-identifier;**
	Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid.
	Note: SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
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=NETEZZA**;
	Specifies the data service for the Netezza database to which you want to connect.
	Note: You must specify the driver.
CONNECTION OPTIONS	**CONOPTS=(ODBC—compliant database connection string);**
	Specifies an ODBC-compliant database connection string using ODBC-style syntax. These options, combined with the ODBC_DSN option, must specify a complete connection string to the data source. If you include a DSN= or FILEDSN= specification within the CONOPTS= option, do not use the ODBC_DSN= connection option. However, you can specify the ODBC database-specific connection options by using CONOPTS=. Then you can specify an ODBC DSN that contains other connection information by using the ODBC_DSN= connection option.

Option	Description
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 Netezza database resides.
PORT	**PORT=port_number** Identifies the listen port of the server where the Netezza database resides.

Advanced Connection Options

The Driver for Netezza supports the following advanced connection options.

Option	Description
CLIENT ENCODING	**CLIENT_ENCODING=cei** Used to specify encoding for the client.
CT_PRESERVE	**CT_PRESERVE=STRICT │ SAFE │ FORCE │ FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.

Option	Description
DEFAULT_ATTR	**`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: • **`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. <table><tr><td>0</td><td>Causes the driver to use client-side static cursor emulation if a scrollable cursor is requested but the database server cannot provide one.</td></tr><tr><td>1</td><td>Causes the driver to always use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is not used.</td></tr><tr><td>2</td><td>(Default) Causes the driver to never use client-side static cursor emulation if a scrollable cursor is requested. The database server's native cursor is used if available. Otherwise, the cursor will be forward-only.</td></tr></table> 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). 0 is the default. 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 as follows: • **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 is 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.
USER ID	**USER="user-id";** Specifies a Netezza user ID. If the ID contains blanks or national characters, enclose it in quotation marks. Alias: UID. *Note:* You must specify the USER option.
PASSWORD	**PASSWORD=password;** Specifies a password for the ID passed through the USER= option. Alias: PWD. *Note:* You must specify the PASSWORD option with USER.
SCHEMA	**SCHEMA=schema-name** Specifies a schema name that overrides the default schema. Multiple schemas are supported with Netezza 7.0.3 or later. Additional schemas can be specified in FedSQL by qualifying the table name. Alias: SCHEMANAME.
STRIP_BLANKS	**STRIP_BLANKS=YES│NO;** Specifies whether to strip blanks from character fields.
READ ONLY	**READONLY=YES│NO;** Specifies whether to connect to the Netezza database in Read-Only mode. The default is NO. Alias: READ_ONLY
SHOW SYSTEM TABLES	**SHOWSYSTEMTABLES=YES│NO;** Specifies whether tables are included in the available table list. If set to YES or TRUE, system tables are included in the available table list. The default setting is NO. Alias: SST
NUMBER BYTES PER CHARACTER	**NUMBYTESPERCHAR=value;** Specifies the default number of bytes per character.

SAS Federation Server Driver for ODBC

Overview

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 (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.

About the SAS Federation Server Driver for ODBC

The SAS Federation Server Driver for ODBC (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 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 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.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See Setting Environment Variables for additional information.

Data Service Connection Options for ODBC

Overview

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Connection Options

The Driver for ODBC supports the following connection options.

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 databases like SQL Server that do support native catalogs, 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 database. For example, to map the logical catalog **mycat** to the native catalog named **newusers**, use the following command: **catalog=(mycat=newusers);**. Catalog name maps can be used only with FedSQL. They are not valid with native SQL. *Note:* SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
CONOPTS	**CONOPTS=(ODBC—compliant database connection string);** Specifies an ODBC-compliant database connection string using ODBC-style syntax. These options, combined with the ODBC_DSN option, must specify a complete connection string to the data source. If you include a DSN= or FILEDSN= specification within the CONOPTS= option, do not use the ODBC_DSN= connection option. However, you can specify the ODBC database-specific connection options by using CONOPTS=. Then you can specify an ODBC DSN that contains other connection information by using the ODBC_DSN= connection option. Here is an example string using the CONOPTS option: **DRIVER=ODBC;CONOPTS=(DRIVER={DataFlux 32-bit SQL Server Wire Protocol};SERVER=myserver;APP=Microsoft ODBC SDK;DATABASE=mydb)** This example uses the ODBC_DSN option with the CONOPTS option: **DRIVER=ODBC;ODBC_DSN=mydsn;CONOPTS=(DATABASE=mydb)**
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:* DRIVER is a required option. 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 Connection Options

The Driver for ODBC supports the following advanced connection options for an ODBC-compliant database.

Option	Description
CLIENT_ENCODING	**CLIENT_ENCODING=encoding-value** Specifies a client encoding value that overrides the default. The default is the encoding that is set on the machine on which SAS Federation Server is running.

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 is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.
ENABLE MULTIPLE ACTIVE RESULT SETS (MARS)	**ENABLE_MARS= NO│YES** 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 YES which gives FedSQL the ability to allow transactions under a given SQL Server connection.
DEFAULT_ATTR	**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: • **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.

For the CURSORS values:

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 is not 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 is 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). 0 is the default. 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. *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
DM_UNICODE	**DM_UNICODE=unicode-setting** Specifies the Unicode setting for the Driver Manager. The default is UTF-8 which is the requirement for the SAS and DataFlux branded drivers. Use **DM_UNICODE=UCS2** to connect to unixODBC based drivers so that the correct Unicode setting is realized. See "Configuring ODBC Connections Using Third Party ODBC Drivers" on page 34 for additional information.

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 as follows: • **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 is 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.
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=**.

Option	Description
PASSWORD	**PASSWORD=password;** Specifies the password that corresponds to the user ID in the database. *Note:* The alias is **PWD=**.

Here are example connection strings using the SAS Federation Server Driver for ODBC:

This connection string specifies an ODBC DSN:

```
driver=odbc; uid=scott; pw=mypw; 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;
```

Configuring ODBC for Hadoop

Connection Options

In addition to the ODBC data service connection options above, the following option is available for Hive and Impala using ODBC:

Option	Description

SCHEMA	**SCHEMA=schema-name;**
	
	This option is for use with Apache Hive or Cloudera Impala only. Specifies the name of the schema that is passed to FedSQL.
	
	This example connection string specifies a schema name that is passed to FedSQL for Hive or Impala connections:
	
	`DRIVER=ODBC;UID=hadoop;PWD=mypw;CONOPTS=`
	`(DSN=tktshive);SCHEMA=schema1;CATALOG=CATALOG_HIVE;`

See the SAS Federation Server Driver for Apache Hive on page 150 for additional information about configuring ODBC options for Hive.

Wire Protocol Driver Usage Notes

Overview

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.

You cannot use the unixODBC driver manager if the wire protocol drivers are installed.

Note: The behavior of a DSN using a wire protocol driver with the catalog option selected, returns only the schemas that have associated tables or views. To list all existing schemas, create a DSN without the catalog option selected.

MySQL
When configuring an ODBC DSN using the MySQL Wire Protocol driver, select the following advanced options:

- **Application Using Threads**

- **Enable SQLDescribeParam**

SQL Server and SQL Server Legacy
Configure the following **Advanced** options for the SQL Server Wire Protocol driver and the SQL Server Legacy Wire Protocol driver:

- **Application Using Threads**

- **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.

SAS Federation Server Driver for Oracle

About the SAS Federation Server Driver for Oracle

The SAS Federation Server Driver for Oracle (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 Driver for Oracle supports most of the FedSQL functionality. The driver also supports the application's ability to submit native Oracle SQL statements.

The 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.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See Setting Environment Variables for additional information.

Data Service Connection Options for Oracle

Overview

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Connection Options

The Driver for Oracle supports the following connection options.

Option	Description
CATALOG	**CATALOG=catalog—identifier;**
	Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. Any identifier is valid such as **catalog=oracle_test**. You must specify a catalog. For the Oracle database, this is a logical catalog name to use as an SQL catalog identifier.
	Note: SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
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 as defined in tnsnames.ora or other naming method. A connect identifier can be a net service name or a database service name that resolves to a connect descriptor.
	`DRIVER=oracle USERID=myusr1 PASSWORD=mypwd1 PATH=tktsora`
	Connect identifiers used in a connection string cannot contain spaces, unless enclosed within single quotation marks or double quotation marks
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$sysid** is used, if it is enabled.
PASSWORD (PWD)	**PWD=password;**
	Specifies an optional Oracle database password that is associated with the Oracle user ID. **PWD=** is always used with **UID=** and the associated password is case-sensitive. If you omit **PWD=**, the password for the default Oracle user ID **OPS$sysid** is used, if it is active.

Advanced Connection Options

The Driver for Oracle supports the following advanced connection options.

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 is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.
DEFAULT_ATTR	**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: • **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 is not 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 is 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). 0 is the default. 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 as follows: • **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 is 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 that 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 192. 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 the connection option, **USE_CACHED_CATALOG**, is set to YES.

```
/*----------------------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.

You cannot use the unixODBC driver manager if the wire protocol drivers are installed.

Note: When you use a wire protocol driver to create an ODBC connection, the following special considerations apply:

1. The behavior of a DSN using a wire protocol driver with the catalog option selected, returns only the schemas that have associated tables or views. To list all existing schemas, create a DSN without the catalog option selected.

2. Verify that the Enable Bulk Load option is turned on in the ODBC DSN for databases that support this option. The Bulk Load option is not enabled by default in the newer wire protocol drivers. As a result, insert performance suffers.

When configuring an ODBC DSN using the Oracle Wire Protocol driver, set the following advanced options:

- **Application Using Threads**
- **Enable SQLDescribeParam**
- **Describe at Prepare**
- **Enable N-CHAR Support**
- **Enable Scrollable Cursors**

SAS Federation Driver for PostgreSQL

About the SAS Federation Server Driver for PostgreSQL

The SAS Federation Server Driver for PostgreSQL (Driver for PostreSQL) enables SAS Federation Server to read and update legacy PostgreSQL tables. In addition, the driver creates PostgreSQL tables that can be accessed by both SAS Federation Server and the PostgreSQL data management system.

The Driver for PostgreSQL supports most of the FedSQL functionality. The driver also supports an application's ability to submit native SQL statements.

The Driver for PostgreSQL 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 PostgreSQL

Overview
To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

Connection Options
The following connection options are supported for Postgres data sources.

Option	Description
CATALOG	**CATALOG=catalog-identifier;**
	Specifies an arbitrary identifier for an SQL catalog, which groups schemas that are logically related, for example, **catalog=ptgtest**.
	Note: SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
CONOPTS	**CONOPTS=(ODBC—compliant database connection string);**
	Specifies an ODBC-compliant database connection string using ODBC-style syntax. These options, combined with the ODBC_DSN option, must specify a complete connection string to the data source. If you include a DSN= or FILEDSN= specification within the CONOPTS= option, do not use the ODBC_DSN= connection option. However, you can specify the ODBC database-specific connection options by using CONOPTS=. Then you can specify an ODBC DSN that contains other connection information by using the ODBC_DSN= connection option.
	Here is an example string using the CONOPTS option: **DRIVER=ODBC;CONOPTS=(DRIVER={DataFlux 32-bit SQL Server Wire Protocol};SERVER=myserver;APP=Microsoft ODBC SDK;DATABASE=mydb)**
	This example uses the ODBC_DSN option with the CONOPTS option: **DRIVER=ODBC;ODBC_DSN=dsn-name;CONOPTS=(DATABASE=database-name)**
DRIVER	**DRIVER=POSTGRES**;
	Specifies the data service for the PostgreSQL database to which you want to connect.
	Note: Using **DRIVER=POSTGRES** on UNIX requires the "unixODBC Driver Manager".
DATABASE	**DATABASE=database-name;**
	Specifies the name of the PostgreSQL database. Enclose the database name in single quotation marks if it contains spaces or non-alphanumeric characters. You can also specify DATABASE= with the DB= alias.**database=sample, DB=sample**.
POSTGRES_DSN	**PTG_DSN=data-source-name;**
	Specifies the data source name to which you want to connect. Alias: PTG_DSN

Option	Description
PASSWORD	**PWD=password;**
	Specifies the password associated with the user ID. Enclose password in single quotation marks if it contains spaces or nonalphanumeric characters. You can also specify PASSWORD= with the PWD=, PASS=, and PW= aliases.
PORT	**PORT=port_number**
	Specifies the port number that is used to connect to the specified PostgreSQL Server. If you do not specify a port, the default is 5432.
SERVER	**SERVER='server-name'**
	Specifies the server name or IP address of the PostgreSQL server to which you want to connect. Enclose the server name in single quotation marks if the name contains spaces or non-alphanumeric characters: **SERVER='server name'**.
USER	**USER=user-name**
	Specifies the PostgreSQL user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or non-alphanumeric characters, you must enclose it in single quotation marks.

Advanced Options

The following advanced options are supported for PostgreSQL data sources.

Option	Description
ALLOW UNQUOTED NAMES	**ALLOW_UNQUOTED_NAMES=NO│YES**
	Specifies whether to enclose table and column names in quotation marks. Tables and columns are quoted when this option is set at NO. If set to YES, the driver does not automatically add quotation marks to table and column names if they are not specified. This allows PostgreSQL tables and columns to be created in the default lowercase. The default option is NO.
CLIENT ENCODING	**CLIENT_ENCODING=cei**
	Used to specify encoding for the client.

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 is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.
DEFAULT_ATTR	**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: • **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 is not 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 is 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). 0 is the default. 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 as follows: • **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 is 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, in bytes, that limits the length of long binary fields (LONG VARBINARY). Unlike other databases, PostgreSQL does not have a size limit for long binary fields. The default is 1048576.

Option	Description
MAX_CHAR_LEN	**MAX_CHAR_LEN=value;** Specifies a value that limits the length of character fields (CHAR and VARCHAR). The default is 2000.
MAX_TEXT_LEN	**MAX_TEXT_LEN=value;** Specifies a value that limits the length of long character fields (LONG VARCHAR). The default is 409500.
SCHEMA	**SCHEMA=value;** Specifies the default schema for the connection. If not specified, the schema, or list of schemas, is determined based on the value of the schema search path defined on the database server.
STRIP_BLANKS	**STRIP_BLANKS=YES | NO;** Specifies whether to strip blanks from character fields.

SAS Federation Server Driver for SAP

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.

Prerequisites

SAP software requires extensive configuration before it can be used. See "Installing and Configuring the SAS Federation Server Driver for SAP" for additional information.

Data Service Connection Options for SAP

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

The table below describes the data service connection options for the SAS Federation Server Driver for SAP.

Option	Description
ABAPFM	**ABAPFM = abap_function_name** Specifies the name of the Advanced Business Application Programming (ABAP) function module that the driver uses internally. The default is **/SAS/Z_SAS_DIALOG**. Alias: ABAPFUNCTION, ABAPFUNC
ABAP_NAMESPACE	**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. The default is **/SAS/** Alias: ABAPNAMESPACE, ABAP_NAME_SPACE, ABAPNS, ABAP_NS
ABAPPROG	**ABAPPROG = abap_program** Specifies the name of the ABAP language that the driver uses internally. This value is set by the ABAP function module. The default is **/SAS/Z_SAS_READ**. Alias: ABAPREPORT, ABAPPROGRAM
ASHOST	**ASHOST=application_server_host** Specifies the host name of the server or IP address of a specific application server. There is no default. Alias: HST, RFCHOST, R3HOST
BATCH	**BATCH = 0 | 1 | Y | N** Specifies whether the SAS Federation Server Driver for SAP should use SAP batch jobs for the data extracts. If set at **Y** (Yes), the SAS Federation Server Driver for SAP uses batch jobs to extract R/3 data. When set at **N** (No), the SAS Federation Server Driver for SAP uses dialog processes to extract R/3 data. The default is **N** (No). Alias: BATCH_MODE, BATCHMODE
BUFFER_SIZE	**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. The default is 100,000 bytes. Alias: BUFFERSIZE, BUFFSIZE, BLOCK_SIZE, BLOCKSIZE
CLIENT	**CLIENT=client_number** Specifies the SAP logon parameter client. Examples for a client are 000 or 800. The default is the SAP system default. *Note:* When you access the SAP system using the driver, specify valid logon information including **CLIENT**, **USER NAME**, **PASSWORD**, and **LANGUAGE**. The user ID and password can also be passed through single sign-on (SSO). The driver performs a logon check at OPEN time. Alias: CLI, RFCCLIENT, RFCCLI

Option	Description

DEFAULT_ATTR	**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:
	
	• **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 is not 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 is 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). 0 is the default. Example: **DEFAULT_ATTR=(XCODE_WARN=1)**

DESTGROUP	**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. The default is **SAS1**.
DESTINATION	**DESTINATION=destination** Specifies the destination in the sapnwrfc.ini file, if working with the NetWeaver RFC library and a sapnwrfc.ini file. By default the NW RFC library looks for the sapnwrfc.ini file in the current working directory of the process. You can define the path to sapnwrfc.ini by setting the RFC_INI environment variable. When setting the RFC_INI environment variable, specify only the path to the directory that holds sapnwrfc.ini. Do not append the path with the filename. Alias: DEST, DST, DSTN
GROUP	**GROUP=application_server_group** Specifies the name of the group of application servers for load balancing.
GWHOST	**GWHOST = gateway_host_name** Specifies the host name of the SAP gateway, if the server is R/2 or external. Alias: GATEWAY_HOST

Option	Description
GWSERV	**GWSERV = gateway_service** Specifies the service of the SAP gateway, if the server is an R/2 server or external. Alias: GATEWAY_SERVICE
IEEE_REVERSE	**IEEE_REVERSE = Y | N** Specifies whether floating point numbers are byte reversed. The possible values are Y (floating-point numbers are byte reversed) and N (floating-point numbers are not byte reversed). The default value for an R/3 application server on Windows NT is Y. On other platforms, the default value is N.
INENCODING	**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.
LANGUAGE	**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. If not specified, the language set on the SAP system is the default. *Note:* When you access the SAP system using the driver, specify valid logon information including **CLIENT**, **USER NAME**, **PASSWORD**, and **LANGUAGE**. The user ID and password can also be passed through single sign-on (SSO). The driver performs a logon check at OPEN time. Alias: LANG, LNG, RFCLANG, RFCLNG
MAX_TABLE_JOINS	**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. The default is 25. Alias: MAX_TABLES_JOIN, MAX_TABLES_JOINS, MAX_TABLE_JOIN
MSHOST	**MSHOST = message_server_host** Specifies the host name of the Message Server for load balancing.
NAMESPACE	**NAMESPACE=namespace** Specifies the namespace to be viewed with directory services. Also used to apply table filters that limit the number of SAP namespaces returned by the SAS Federation Server Driver for SAP by specifying multiple items in a comma-delimited string: **NAMESPACE={T000, AAA, B1}**.
PWD	**PWD=password** Specifies the SAP logon parameter password. *Note:* When you access the SAP system using the driver, specify valid logon information including **CLIENT**, **USER NAME**, **PASSWORD**, and **LANGUAGE**. The user ID and password can also be passed through single sign-on (SSO). The driver performs a logon check at OPEN time. Alias: PASSWORD, PASSWD, PW, PASS

Option	Description
R3NAME	**R3NAME=system_name** Specifies the name of the R/3 system, if load balancing is being used.
RFC_STRING	**RFC_STRING = additional_rfc_options** Specifies additional logon or connection parameters for the RfcOpenConnection() call. The SAS Federation Server Driver for SAP uses the RfcOpenConnection() call to log on to the SAP system. Using this option, parameters that are not connection attributes for the SAS Federation Server Driver for SAP, can be passed to the RfcOpenConnection() call. The parameters are passed as name value pairs, for example: **RFC_STRING = "ABAP_DEBUG=1"**. Alias: RFCSTRING, RFC_OPTIONS_EXT, RFCOPENEX, ADDITIONAL_RFC_OPTIONS
SAPLOGON_ID	**SAPLOGON_ID = saplogon_id** Specifies the string defined for SAPLOGON on a Windows 32-bit system. SAPLOGON_ID is not supported if working with the NetWeaver RFC library.
SYSNR	**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. Alias: SYS, SYSTEM, SYSNO
TRACE	**TRACE = 0 | 1 | Y | N** Specifies if the SAS Federation Server Driver for SAP should trace requests. If the trace option is set to **1** or **Y**, the driver writes log information into a file. The RFC library logs messages in the **dev_rfc** file. The default setting is **0** (**N**), RFC trace is off. *Note:* The RFC trace directory is set using the **RFC_TRACE_DIR** environment variable.
UID	**UID = user** Specifies the SAP logon parameter user. *Note:* When you access the SAP system using the driver, specify valid logon information including **CLIENT**, **USER NAME**, **PASSWORD**, and **LANGUAGE**. The user ID and password can also be passed through single sign-on (SSO). The driver performs a logon check at OPEN time. Alias: USR, USER, RFCUSER, USERNAME, USERID

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.

- 64-bit SAP Unicode RFC library, Release 7.20 or higher, or 64-bit SAP NetWeaver RFC library, Release 7.20 or higher.

- The NetWeaver RFC library requires NW RFC SDK 7.20, patch level 36 or higher.

The SAS Federation Server Driver for SAP for Windows and UNIX requires the SAP NetWeaver RFC Library, Release 7.20, which is provided by SAP AG.

As of the end of maintenance for SAP Release 7.10 (March 31, 2016), SAP no longer supports the classic RFC SDK or the classic RFC library. This end of maintenance also applies to SAP Releases 7.11 and 7.20. A transition to the SAP NetWeaver RFC Library should start immediately. The SAP NetWeaver RFC Library supports all SAP NetWeaver and R/3 systems and supports Unicode and non-Unicode. Refer to SAP note 1025361 for installation instructions, support information, and details about the availability of the SAP NetWeaver Library.

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 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] Only required 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 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 Netweaver RFC libraries from SAP.

   The Driver for SAP requires the 64-bit version of the SAP NetWeaver RFC Library. This library must be installed on SAS Federation Server.

   SAP no longer supports the classic RFC SDK or the classic RFC library after the maintenance end of SAP Release 7.10 (March 31, 2016). This end of maintenance also applies to SAP Releases 7.11 and 7.20. Users of the Driver for SAP should immediately start to transition to the SAP NetWeaver RFC Library.

   The SAP NetWeaver RFC Library supports all SAP NetWeaver and R/3 systems and supports Unicode and non-Unicode. Refer to SAP note 1025361 for information

about installation, support, and availability of the SAP NetWeaver Library. If necessary, refer to SAP Note 413708 for information about the classic version of the RFC Library.

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 10.2 *AIX*

Bourne Shell	`$ LIBPATH=rfclib_directory:$LIBPATH` `$ export LIBPATH`
C Shell	`$ setenv LIBPATH rfclib_directory:$LIBPATH`

   Table 10.3 *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 10.4 *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 10.5 *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 install the SAPGUI 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)	SAPKA94030INSAS *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
	SAPKA94034INSAS	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 on 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  SAPKA94030INSAS SID
tp import SAPKA94030INSAS 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 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`
	`SAPKA94030INSAS`*SID*
	`pf=\usr\sap\trans\bin\TP_DOMAIN_`*SID*`.PFL`
	`-D"nbufform=true" -D"tp_version=264"`
	
	`tp import`
	`SAPKA94030INSAS`*SID*
	`pf=\usr\sap\trans\bin\TP_DOMAIN_`*SID*`.PFL`
	`-D"nbufform=true" -D"tp_version=264"`

UNIX	`$ tp addtobuffer`
	`SAPKA94030INSAS `*SID*
	`pf=/usr/sap/trans/bin/TP_DOMAIN_`*sid*`.PFL`
	`-D"nbufform=true" -D"tp_version=264"`
	
	`$ tp import`
	`SAPKA94030INSAS`*SID*
	`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`
	`SAPKA94030INSAS`*SID*
	`pf=\usr\sap\trans\bin\TP_DOMAIN_`*SID*`.PFL`
	`-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"`
	
	`tp import`
	`SAPKA94030INSAS`*SID*
	`pf=\usr\sap\trans\bin\TP_DOMAIN_`*SID*`.PFL`
	`-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"`

UNIX	`$ tp addtobuffer`
	`SAPKA94030INSAS`*SID*
	`pf=/usr/sap/trans/bin/TP_DOMAIN_`*SID*`.PFL`
	`-D"nbufform=true" -D"tp_version=264" -D"setunicodeflag=true"`
	
	`$ tp import`
	`SAPKA94030INSAS`*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
SAPKA93130INSASSID
pf=\usr\sap\trans\bin\TP_DOMAIN_SID.PFL
-D"nbufform=true"

tp import
SAPKA93130INSASSID
pf=\usr\sap\trans\bin\TP_DOMAIN_SID.PFL
``` |
| UNIX | ```
$ tp addtobuffer
SAPKA93130INSASSID
pf=/usr/sap/trans/bin/TP_DOMAIN_sid.PFL
-D"nbufform=true"

$ tp import
SAPKA93130INSASSID
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
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 10.6 *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 Federation Server Driver for SAP HANA

About the SAS Federation Server Driver for SAP HANA

The SAS Federation Server Driver for SAP HANA (Driver for SAP HANA) enables Read and Write access to SAP HANA data sources. The driver supports both native SQL and FedSQL dialects.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See Setting Environment Variables for additional information.

Data Service Connection Options for SAP HANA

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

When configuring a data service, you must include one of the following configurations to establish connection to an SAP HANA system:

- SERVER and INSTANCE
- SAPHANA_DSN or a DSN in CONOPTS
- SERVER and PORT
- SERVER with a full server name and port

The following table describes the data service connection options for SAP HANA.

Option	Description
CATALOG	**CATALOG=mysaphanacatalog**
	Specifies an arbitrary identifier for an SQL catalog, which groups logically related schemas. CATALOG is a required option.
	Note: SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
DRIVER	**DRIVER=SAPHANA**
	Identifies the type of data service to which you want to connect. The data service SAPHANA represents the SAP HANA database.

Option	Description
SAPHANA_DSN, DB, DATABASE	Specifies the configured SAP HANA ODBC datasource to which you want to connect. Use this option if you have existing SAP HANA ODBC datasources that are configured on your client. This method requires additional setup, either through the ODBC Administrator control panel on Windows platforms or through the odbc.ini file on UNIX platforms.
	Here is an example of an odbc.ini entry in UNIX:
| | ```
[SAPHANADSN]
SERVERNODE=107.20.242.225:30015
``` |
	Connection options specified in CONOPTS= are appended to the connection string. Use CONOPTS or SAPHANA_DSN to specify the DSN. Do not use both of these options to specify the DSN.
SERVER	Specifies the server name or IP address of the SAP HANA server. The port can be included in the specified value. The port number is 3[instance-number]15 (for example, 30015 for instance number 00).
	You can specify a list of hostnames separated by a semicolon to support failover. If a host is not available, the next host from the list is used.
	alias: SERVERNODE, SERVER, HOST
	Here are some examples using the SERVER= option:
| | ```
SERVER=<'>server-name<'>
SERVER=<'>server-name:port<'>
SERVER='server-name:port;failover-server-name1:port;failover-server-name2:port'
``` |
PORT	**PORT=30015**
	Specifies the port number that is used to connect to the specified SAP HANA server. If you do not specify the port, the instance number, or include the port number in the server specification, the default 30015 is used.
	Note: 3[instance]15 is the port for the standard SQL communication for client access. This is the only port required for client access.
INSTANCE	Specifies the instance number of the SAP HANA database engine. The port number is 3[*instance-number*]15. For example, 30015 is the port number for INSTANCE number 00. If the port number is explicitly specified in either the PORT= or the SERVER= option, the INSTANCE= option is ignored, and a warning is written to the server log.
CONOPTS	**CONOPTS=(ODBC-compliant connection string)**
	Specifies an ODBC-compliant database connection string using ODBC-style syntax. These options, combined with the ODBC_DSN option, must specify a complete connection string to the data source. If you include a DSN= or FILEDSN= specification within the CONOPTS= option, do not use the ODBC_DSN= connection option. However, you can specify the ODBC database-specific connection options by using CONOPTS=. Then you can specify an ODBC DSN that contains other connection information by using the ODBC_DSN= connection option.
UID	**UID='user-ID'**
	Specifies the SAP HANA user name, or user ID that you use to connect to a database. If the user name or ID contains spaces or nonalphanumeric characters, enclose it in quotation marks.

Option	Description
PWD	**PWD='user password'** Specifies the password that is associated with your SAP HANA user name. If the password contains spaces or non-alphanumeric characters, you must enclose it in quotation marks. You can also specify PASSWORD= with the PWD=, PASS=, and PW= aliases.

Secure Sockets Layer (SSL) Connection Options

The Driver for SAP HANA supports SSL. Here are the connection options for SSL.

Option	Description
ENCRYPT	**ENCRYPT=0 │ 1** Used to enable or disable SSL encryption. The default is 0 (NO).
SSLCRYPTOPROVIDER	**SSLCRYPTOPROVIDER=SAPCRYPTO │ OPENSSL │ MSCRYPTO** Specifies the cryptographic library provider for SSL connectivity. Alias: SSLPROVIDER
SSLKEYSTORE	**SSLKEYSTORE='file_path'** Specifies the path to the keystore file that contains the server's private key. If a value is not specified, the ODBC driver uses the default **$HOME/.ssl/key.pem**.
SSLTRUSTSTORE	**SSLTRUSTSTORE='file_path'** Specifies the path to the truststore file that contains the server's certificate. If a value is not specified, the ODBC driver uses the default **$HOME/.ssl/trust.pem**. *Note:* Leave this option empty if you are using the mscrypto cryptographic library.
SSLVALIDATECERTIFICATE	**SSLVALIDATECERTIFICATE=NO │ YES │ 0 │ 1** Set this option to validate the server's certificate. Setting to YES or 1 activates validation. The default is NO or 0. If this option is not specified, the ODBC driver uses the default and does not validate certificates.
SSLHOSTNAMEINCERTIFICATE	**SSLHOSTNAMEINCERTIFICATE='string'** Specifies the host name to use for validation. Use this host name when validating a server's certificate using SSLVALIDATECERTIFICATE. Alias: SSLHOSTNAMEINCERT

Option	Description
SSLCREATESELFSIGNEDCERIFICATE	**SSLCREATESELFSIGNEDCERTIFICATE=NO|YES|0|1** Specifies if a self-signed certificate is created if the keystore cannot be found. If set to YES, a self-signed certificate is created in the event that the keystore is not found. If this option is not specified, the driver uses the default, which is NO. Alias: SSLCREATECERT

Advanced Connection String Options

The Driver for SAP HANA supports the following advanced connection options.

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 is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.

Option	Description
DEFAULT_ATTR	**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:

- **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 is not 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 is 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). 0 is the default. 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 as follows: • **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. Therefore, it 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. *Note:* The application can override this value, but if the application does not explicitly set a cursor type, the value specified in **DEFAULT_CURSOR_TYPE** is in effect.

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 as follows: • **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 is 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.

Option	Description

TABLE TYPE	**TABLE_TYPE=ROW|COLUMN|LOCAL|LOCAL TEMPORARY|GLOBAL|GLOBAL TEMPORARY**
	
	Specifies the default table type when creating tables using FedSQL (CREATE TABLE). This option can be overridden by the TABLE_TYPE table option. If the table store type is not specified in connection options nor in the table options, then the default SAP HANA store type is used.
	
	ROW
	    Creates a table using ROW-based storage in SAP HANA.
	
	COLUMN
	    Creates a table using COLUMN-based storage in SAP HANA.
	
	LOCAL | LOCAL TEMPORARY
	    Creates a local temporary table in SAP HANA. The table definition and data are visible only in the current session.
	
	GLOBAL | GLOBAL TEMPORARY
	    Creates a global temporary table in SAP HANA. The global temporary tables are globally available, and the data is visible only in the current session.

SAS Federation Server Driver for SASHDAT

About the SAS Federation Server Driver for SASHDAT

The SAS Federation Server Driver for SASHDAT (Driver for SASHDAT) is a write-only driver designed for use with Hadoop on a grid host, such as the SAS LASR Analytic Server. SAS LASR Analytic Server integrates with Hadoop by storing SAS data in the Hadoop Distributed File system (HDFS). Using the Driver for SASHDAT, you can write files into HDFS, which makes them available for load to SAS LASR Analytic Server. Because the data volumes in HDFS are usually very large, the driver is not designed to read from HDFS and transfer data back to the client.

The Driver for SASHDAT enables a user to create a table and insert into it multiple times. Data is written when the connection is closed or when the user issues a **COMMIT** or **ROLLBACK**. When a table is created, the Driver for SASHDAT stores the table definition in the connection. Catalog functions return information about the table, but the information is not written to HDFS until the first row is inserted. The table remains open and available for data until one of the following conditions are met:

- The SQL command COMMIT WORK is executed. This command closes and finalizes all tables with inserted data that are open for connection.

- A COMMIT or ROLLBACK is received from the client.

- The client issues a disconnect event at which time all tables with inserted data are closed.

- When the COMMIT= statement option is specified in the connection string, the table is closed when the INSERT statement changes to unprepared. If a user creates a table, prepares an INSERT, and executes it multiple times, the table remains open and available for more data. When the statement changes to unprepared, the table is closed. When a table is closed, data can no longer be written. To change the data in an HDFS table, it must be dropped and re-created.

Connection Options

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

The Driver for SASHDAT supports the following connection options.

Option	Description
CATALOG	**CATALOG=catalog name;** Specifies the catalog name for the connection. *Note:* SAS Federation Server automatically quotes SQL identifiers that do not meet the regular naming convention as defined in the *SAS FedSQL Reference Guide*.
COMMIT	**COMMIT=S | STATEMENT | C | CONNECTION;** Specifies when to close the SASHDAT file. Use **S** to close when the statement is unprepared, or **C** when the connection is disconnected. The default is **C**, to close when the connection is disconnected.
COPIES	**COPIES=number-of-copies;** Specifies how many copies are made when file blocks are written to HDFS. Note that specifying **COPIES=0** is valid and signals the engine that you do not want any replicate copies of the data in HDFS. Defaults for this option depend on the setting for NODIST. The default is **1** when **NODIST=NO** is specified and **2** when **NODIST=YES** is specified.
CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.
DEFAULT SCHEMA	**DEFSCHEMA=schema-name;** Specifies the default schema for the connection. When using the DEFSCHEMA option, the default schema and path must also be specified with **SCHEMA=(NAME=schema-namePATH | PRIMARYPATH=path)** in the connection string.

Option	Description
ENCODING	**ENCODING=SAS-NLS-encoding-identifier;** Specifies the encoding for SASHDAT data and character conversions, both to and from. If not specified, the default encoding is inherited from SAS Federation Server. *Note:* Tables created with the Driver for SASHDAT use the encoding specified in the connection string. If the encoding option is not specified, encoding defaults to the character set associated with the operating system for SAS Federation Server.
HASH	**HASH=Y|YES|N|NO;** Specifies the algorithm that determines the distribution of partitions to nodes of the LASR Analytic Server proxy. The default is **HASH=NO**, which specifies that the distribution scheme depends on a binary tree. **HASH=YES** indicates that the distribution scheme depends on a hash function. As a result, the distribution properties of the partitions are not as balanced, but result in less memory usage. **HASH=YES** is recommended when working with high-cardinality partition keys (in the order of millions of partitions).
HOST	**HOST | SERVER=grid-server-name;** Specifies the name of the grid host that has a running Hadoop NameNode. This option is required in the connection string. There is no default.
INSTALL	**INSTALL=path;** Specifies the path to the TKGrid installation on the grid host. This option is required in the connection string. There is no default.
LOCALE	**LOCALE=SAS-locale-identifier;** Specifies the locale for message text and character conversions, both to and from. The default locale is acquired from the server operating system.
NODIST	**NODIST|INNAMEONLY=Y|YES|N|NO;** Specifies whether to place small tables into HDFS. Use **NODIST=Y** as the mode for placing small tables into HDFS. The default is **N** (No).
PASSWORD	**PWD=alternate-password;** Specifies the password for the alternate user when connecting to the grid host with a running Hadoop NameNode.
SCHEMA	**SCHEMA=(NAME=schema-name PATH|PRIMARYPATH=path);** This option maps a logical schema name to a specific path for the grid host with a running Hadoop NameNode. This option can be specified multiple times in a single connection string to define multiple schemas. At least one schema is required. There is no default if a schema is not specified. However, once specified, the first schema listed in the connection string is designated as the default schema if **DEFSCHEMA=** is not used.
SQUEEZE	**SQUEEZE=Y|YES|N|NO** Specifies whether the SASHDAT file will be compressed. The default is N (no compression).
TIMEOUT	**TIMEOUT=timeout-in-seconds;** Specifies the amount of time to wait while trying to establish a connection before terminating the attempt and generating an error. The default is 20 seconds.

Option	Description
USER	**UID=alternate-userid;** Specifies the ID of an alternate user when connecting to the grid host with a running Hadoop NameNode.

Example Connection Strings

The following connection string connects a default user and closes files on disconnect, commit, or rollback:

```
CATALOG=HDAT;DRIVER=SASHDAT;HOST=hostname;INSTALL="/opt/TKGrid/v940m1/laxnd";
  SCHEMA=(name=SCHEMA1;PATH="/user/test");
```

This connection string connects a test user, defines two schemas, stores data in UTF8, and closes files on statement unprepare, disconnect, commit, or rollback:

```
CATALOG=HDAT;DRIVER=SASHDAT;COMMIT=S;UID=test;
  ENCODING=UTF8;HOST=hostname;INSTALL="/opt/TKGrid/v940m1/laxnd/";
  SCHEMA=(name=CUSTOMERS;PATH="/user/custs");SCHEMA=(name=Accounts;PATH="/user/accts");
```

SAS Federation Server Driver for SAS Scalable Performance Data (SPD) Server

About the Driver

With the SAS Federation Server Driver for SPD Server (Driver for SPD Server), SPD Server tables can be accessed for reading, writing, and updating by SAS Federation Server. The Driver for SPD Server provides connectivity from a SAS Federation Server on any host to an active SPD Server. The Driver for SPD Server integrates with UNIX hosts as well as Windows.

Prerequisites

Install SAS SPD Server 5.3 and all hotfixes (including SAS Federation Server, SAS Federation Server Manager, SAS Drivers for Federation Server, and SAS 9.4) before using the Driver for SPD Server.

Securing the Connection with SSL

SAS SPD Server supports Secure Sockets Layer (SSL) connections. The Driver for SPD Server supports SSL to substantiate a secure connection. To activate SSL, you must set the SPDSRSSL environment variable. For UNIX installations, you must define the path to the client certificate with the SSLCALISTLOC environment variable. Here is an example:

```
export SPDSRSSL=YES
export SSLCALISTLOC=/u/dohaig/pp/OpenSSL/certs/demoCA-RSA2048-SHA384.pem
```

SPDSRSSL

```
SPDSRSSL=NO|YES|PREFERRED
```

The default value is NO, which specifies that client connections are not secured. To enable SSL, specify YES or PREFERRED. YES specifies that the server requires a secure client connection. PREFERRED specifies that a secure connection is allowed but not required. If a client is configured for SSL, the connection is secured.

SSLCALISTLOC

```
export SSLCALISTLOC=/opt/certs/demoCA-RSA2048-SHA384.pem
```

Specifies the location of the public certificate or certificates for trusted certificate authorities (CA).

Note: This environment variable applies to UNIX installations only.

You can also set these environment variables using the SetEnv option set in the **dfs.serv.common.xml** configuration file:

```
<OptionSet name="SetEnv">
    <Option name="SPDSRSSL">YES</Option>
    <Option name="SSLCALISTLOC">/opt/certs/demoCA-RSA2048-SHA384.pem</Option>
</OptionSet>
```

When you use the SetEnv option set, the environment variables are automatically defined each time SAS Federation Server initializes.

See the *SAS Scalable Performance Data Server: Administrator's Guide* for information about configuring SSL for client authentication.

Data Service Connection Options for SPD Server

Overview

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

To configure a data service for SPD Server, use the GENERIC data service type in a DDL statement. If you are using SAS Federation Server Manager to create a connection for SPD Server, use the **Generic, without native catalog** data service template. See the *SAS Federation Server Manager: User's Guide* for details.

Connection Options

The following tables reflect the required and advanced (optional) connection options for the Driver for SPD Server.

Table 10.7 *Required Connection Options*

Option	Description
CATALOG	**CATALOG='catalog-name'** Required. Specifies the catalog name.
DBQ	**DBQ='SPD-Server-domain-name'** Required. Specifies the domain name. For SPD Server, a domain represents a specific directory of file storage locations. Data access is facilitated by using domains and a name server.

Option	Description
DRIVER	**DRIVER=SPDS**; Required. Specifies the SAS Federation Server driver for SPD Server.
HOST	**HOST=host-name** Required. Specifies the SPD Server, or host name, to which you are connecting.
LOCALE	**LOCALE=locale** Required. Specifies the locale for the connection.
PASSWORD	**PASSWORD=password** Required for the **USER=** option. Specifies a case-sensitive password that is associated with the user ID passed to the SPD Server. Alias: PWD, PASSWD
SERV	**SERV='port-number'** Required. Specifies the port number associated with the server, or host passed through the HOST option. Alias: PORT
USER	**USER=user-name** Required. Specifies a user name that is registered in the server's password database. Alias: UID

Table 10.8 Advanced Connection Options

Option	Description
ACLGROUP	**ACLGRP=group-name** Specifies an ACL group that the server administrator assigned to your server user ID. By default, the server connection uses the first ACL group in your group list to make a connection. Use the **ACLGRP=** option to specify a different group name from the list.
ACLSPECIAL	**ACLSPECIAL=YES│NO** Invokes special access to all server resources in the domain. These privileges override normal ACL restrictions and enable the user to access the resources of other users, as well as to create or modify ACLs of other users. NO is the default value.

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 as follows: • **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 is 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.
ENCODING	**ENCODING=encoding_CEI** Specifies the encoding used for the session.
LOCKING	**LOCKING=YES│NO** Enables record-level locking. NO is the default value.

Option	Description

SCHEMA	**SCHEMA=(NAME='schema-name';DBQ='domain-name');**
	
	SCHEMA=(host='host-name';SERV='port-number';NAME='schema-name';DBQ='domain-name');
	
	Note: The maximum length for a schema name is 8 characters.
	
	Specifies schema name, location, and SPD Server override using the following suboptions:
	
	NAME=
	   Required with the **SCHEMA=** connection option. Specifies the schema name. You can specify multiple schemas. If multiple schemas are specified, the driver treats each **SCHEMA=** option as an additional schema location available under the specified catalog.
	
	DBQ=
	   Required with the **SCHEMA=** connection option. Specifies a domain name. For SPD Server, a domain represents a specific directory of file storage locations. Data access is facilitated by using domains and a name server.
	
	HOST=
	   Specifies the host name of an SPD Server that overrides the parent HOST name and port number for a schema. Since there can be only one parent host for SPD Server, you can use the SCHEMA option to define a secondary SPD Server with a different HOST and port.
	
	SERV=
	   Required with the HOST= connection option. Specifies the port number associated with the server configured in the HOST option.

This example connection string connects to SPD Server, lax94d01, running on port 14512. It connects to two SPD Server domains: the PUBLIC domain, referenced by the SCHEMA1 schema name; and the FORMATS domain, referenced by the SCHEMA2 schema name:

```
driver=spds;dbq='PUBLIC';host='lax94d01';serv='14512';
locale='en_us';catalog=cat1;SCHEMA=(name='SCHEMA1';dbq='public');
SCHEMA=(name='SCHEMA2';dbq='formats')
```

SAS Federation Server Driver for Teradata

About the SAS Federation Server Driver for Teradata

The SAS Federation Server Driver for Teradata (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 driver supports Teradata client 14 which allows naming up to 32 characters.

The Driver for Teradata supports most of the FedSQL functionality. The driver also supports an application's ability to submit native Teradata SQL statements.

The 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.

Prerequisites

Before configuring SAS Federation Server drivers, you must set environment variables that point to the client libraries required for your data source. See Setting Environment Variables for additional information.

Data Service Connection Options for Teradata

Connection Options

To access data that is hosted on SAS Federation Server, a client must submit a DSN that defines how to connect to the data. DSNs are associated with a data service that provides the foundation for the connection, such as user access control. See "Working with Data Services" for additional information.

The Driver for Teradata supports the following connection options 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=TERADATA;**
	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 Connection Options

The Driver for Teradata supports the following advanced options for a Teradata database.

Option	Description
ACCOUNT	**ACCOUNT=account-ID;**
	Specifies an optional account number that you want to charge for the Teradata session.

Option	Description

CLIENT ENCODING	**CLIENT_ENCODING=encoding-value**
	
	Used to specify the character set for the session. UTF8 is the default if encoding is not specified. The character sets supported are:
	
	`ASCII`
	`EBCDIC`
	`EBCDIC273_0E`
	`EBCDIC277_0E`
	`EBCDIC037_0E`
	`KATAKANAEBCDIC`
	`KANJIEUC_0U`
	`LATIN9_0A`
	`THAI874_4A0`
	`LATIN1250_1A0`
	`CYRILLIC1251_2A0`
	`LATIN1254_7A0`
	`HEBREW1255_5A0`
	`ARABIC1256_6A0`
	`LATIN1258_8A0`
	`TCHBIG5_1R0`
	`SCHINESE936_6R0`
	`KANJI932_1S0`
	`HANGUL949_7R0`
	`TCHINESE950_8R0`
	`LATIN1252_3A0`
	`SCHEBCDIC935_2IJ`
	`TCHEBCDIC937_3IB`
	`HANGULEBCDIC933_1II`
	`KANJIEBCDIC5035_0I`
	`KANJIEBCDIC5026_0I`
	`UTF8`
	`UTF16`

CT_PRESERVE	**CT_PRESERVE = STRICT | SAFE | FORCE | FORCE_COL_SIZE** Allows users to control how data types are mapped. Note that data type mapping is 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 as follows: • **STRICT** The requested type must exist in the target database. No type promotion occurs. If the type does not exist, an error is returned. • **SAFE** Target data types are upscaled only if they do not result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is chosen, even if it could potentially result in a loss of precision or scale. When character encodings are changed, the new column size is 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 is the same as the original encoding. This option can be used to avoid column size creep. However, the resulting column might be too large or too small for the target data.

Option	Description
DEFAULT_ATTR	**`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: • **`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 is not 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 is 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). 0 is the default. 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 as follows: • **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 is 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.
MAXPARMSIZE	**MAXPARMSIZE=size-in-bytes** Specifies the maximum byte limit for parameter bindings for variable length data types (VARCHAR, CHAR, VARBINARY, BINARY). Use this connection option if the number of required parameters exceeds the driver's limit of 64,256 bytes. The default value is 8K (8192 bytes). Alias: MPS
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.
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

Administration DDL

Description

Administration DDL is a set of commands used for administration of SAS Federation Server. Most administration DDL is performed by an administrator (defined as a user who has the ADMINISTER privilege on SAS Federation Server), but some commands, such as CREATE CACHE, require specific privileges which can be assigned to users and groups.

GENERIC Options

Generic options is a set of comma separated name or name-value pairs, that can be used within an OPTIONS list. When a DDL statement contains an options list, you can define additional configurations using GENERIC options.

```
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
```

Use ALTER GENERIC options to include a modifier for a specific option. Modifiers such as ADD, SET, XSET or DROP can be used when altering system objects, for example a SCHEMA. Here is an example:

```
ALTER SCHEMA "catalog1_BASE"."schema3_BASE" {OPTIONS add conopts (LOCKTABLE SHARE)}
```

```
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
```

alter-operation
  Indicates the required action for the specified options. If a value is omitted for alter-operation, ADD is the default modifier. The possible values are:

  ADD
    Adds the specified option.

SET
>   Changes an option that already exists.

XSET
>   Sets the option if it has already been added. Otherwise, adds the option if it does not already exist.

DROP
>   Drops the specified option.

User and Object Privileges

GRANT and DENY

Overview

Use the GRANT or DENY statements to control access to objects that are managed by SAS Federation Server. You can specify what server objects can run an operation and indicate the users and groups that are authorized to access them. When submitting a GRANT, DENY, or REVOKE statement, objects must be fully qualified, with each object surrounded in double quotation marks. In the example that follows, TABLE1 is fully qualified with catalog and schema name parts:

```
GRANT SELECT ON "CAT1"."SCHEMA1"."TABLE1" TO "Bob"
```

Merely specifying **GRANT SELECT ON TABLE1 to "Bob"** produces an error due to missing objects because the table object is not fully qualified.

```
GRANT SELECT ON TABLE CAT1.SCHEMA1.TABLE1 TO BOB
```

Fully qualified catalog names are one-part names:

```
GRANT SELECT ON CATALOG "CAT1" TO "Bob"
```

Fully qualified data service and DSN names are also one-part names:

```
GRANT CONNECT ON DATA SERVICE "BASE" TO "Jane", "Tarzan"
GRANT CONNECT ON DSN "DSN_PUBLIC" TO PUBLIC
```

Fully qualified schema names are two-part names:

```
GRANT SELECT ON SCHEMA "CAT1"."SCHEMA1" TO "Bob"
```

Note: You cannot GRANT or DENY the CREATE DSN or ADMINISTER privileges to the PUBLIC or SASUSERS group.

Syntax

The GRANT and DENY statement specification is summarized below.

Server
```
GRANT | DENY {<server-privilege> [,... <server-privilege>]} |
              ALL [PRIVILEGES]}
[ON SERVER]
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Data Service
```
GRANT | DENY {<data-service-privilege> [,... <data-service-privilege>]} |
              ALL [PRIVILEGES]}
```

```
ON [DATA] SERVICE data-service-name
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

DSN

```
GRANT | DENY CONNECT
              | ALL [PRIVILEGES]}
ON DSN dsn-name
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Catalog

```
GRANT | DENY {<container-privilege> [,... <container-privilege>]}
              | ALL [PRIVILEGES]}
ON CATALOG catalog-name
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Schema

```
GRANT | DENY {<container-privilege> [,... <container-privilege>]}
              | ALL [PRIVILEGES]}
ON SCHEMA catalog-name
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Package or Thread (DS2)

```
GRANT | DENY EXECUTE
              | ALL [PRIVILEGES]}
ON PACKAGE | THREAD object-name
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Table

```
GRANT | DENY {<object-privilege> [,... <object-privilege>]}
              | ALL [PRIVILEGES]}
ON [TABLE] table-name
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Table / Fine Grained Access (RLS)

```
GRANT SELECT
ON [TABLE] table-name
TO {<identity> [,... <identity>]}
WHERE <rls-condition>
[AS ADMINISTRATOR]
```

Column

```
GRANT | DENY {<column-privilege> [,... <column-privilege>]}
              | ALL [PRIVILEGES]}
ON table-name ( column-name, [,... column-name] )
TO {<identity> [,... <identity>]}
[AS ADMINISTRATOR]
```

Arguments

<server-privilege> **: :=**

```
ADMINISTER
| TRACE
| <data-service-privilege>
```

<data-service-privilege> **: :=**
```
| CREATE DSN
| CONNECT
| <container-privilege>
```

<container-privilege> **: :=**
```
| CREATE TABLE
| ALTER TABLE
| CREATE VIEW
| ALTER VIEW
| DROP VIEW
| CREATE CACHE
| ALTER CACHE
| CREATE TABLESPACE
| <object-privilege>
```

<object-privilege> **: :=**
```
| EXECUTE
| DELETE
| <column-privilege>
```

<column-privilege> **: :=**
```
| SELECT
| INSERT
| UPDATE
| REFERENCES
```

<identity>
Specifies a user, user-defined group, or the SASUSERS or PUBLIC group.

<server-privilege>
Specifies a server, a data service, schema-, object-, or column-level privilege.

<data-service-privilege>
Specifies a data service, schema-, object-, or column-level privilege.

<container-privilege>
Specifies a schema-, object-, or column-level privilege.

<object-privilege>
Specifies an object- or column-level privilege.

<column-privilege>
Specifies a column-level privilege.

[WHERE *<rls-condition>*]
`WHERE` clause specifying a row-level security or fine-grained access filter.

identity
An SQL identifier that specifies a user, user-defined group, or the SASUSERS or PUBLIC group.

catalog
Catalog name, an SQL identifier.

schema
Schema specification, a two-part SQL identifier.

table
Table specification, a two-part SQL identifier.

column
Column name, an SQL identifier.

[AS ADMINISTRATOR]
> Grants privileges using the ADMINISTRATOR role. If this option is not used, the privilege is granted as the individual user. If the user is a system user, privileges are assigned with SYSTEM as the grantor.

REVOKE

Use a REVOKE statement to remove explicitly granted or denied privileges from the specified object. Syntax for the REVOKE and GRANT statement is almost identical except that the TO keyword of the TO clause is replaced with a FROM keyword instead. One additional difference is that the WHERE clause does not apply to `revoke SELECT` statements.

Here is an example REVOKE statement:

```
REVOKE SELECT from "user1"
```

DROP AUTHID

Description
Use the DROP AUTHID statement to either drop from the system tables all references to a user or group identity or transfer them to another identity. The dropped or transferred user or group identity is specified using the external authorization identifier rather than the name. External user and group identifiers are displayed in the AUTHORIZATION IDENTIFIERS information view.

Parameters
```
DROP { AUTHID | AUTHORIZATION [IDENTIFIER] } id [TRANSFER TO name]
     [CASCADE|RESTRICT] [FORCE]
```

AUTHORIZATION IDENTIFIER *id*
> Specifies the external identifier of the identity to drop. The `id` specification must be surrounded in double quotation marks. This identifier is displayed in the AUTHORIZATION_IDENTIFIERS information view.

TRANSFER TO *name*
> Specifies the name of the identity that receives object ownership and privileges from the dropped identity. The name specification is an SQL identifier.

> *Note:* The associated identity is created beforehand using SAS Metadata Server.

> CASCADE
>> Specifies that references in system tables to the dropped identifier are removed unconditionally. This option is invalid if the TRANSFER TO option is specified.

> RESTRICT
>> Causes the statement to fail if the dropped identity is the grantor of any privilege or the owner of a DSN or schema. This option is ignored if the TRANSFER TO option is specified.

> FORCE
>> Causes the suppression of error messages when the specified external identifier is not present in any system tables.

Examples
Here are examples of the DROP AUTHID statement:

```
drop authid "A5C2WBEE.A5000JBJ" transfer to user2

drop authid "A5C2WBEE.AP0007PX" cascade
```

Server Configuration

ALTER SERVER

Description

Use ALTER SERVER to change the server configuration by specifying server options. You must be the server owner (SYSTEM account) or administrator to run this statement.

Parameters

```
ALTER SERVER
    [ alter-server-options ]
```

alter-server-options

Specifies the list of server options to alter.

```
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
```

alter-operation

Indicates the required action for the specified options. **ADD** is the default operation if a value is not set. The possible values are:

ADD

Use **ADD** to add the specified option.

SET

Use **SET** to change an option that already exists.

XSET

Use **XSET** to set an option that has previously been added. Otherwise, **XSET** adds the option if it does not exist.

DROP

Use **DROP** to drop the specified option.

server-option

Specifies the server configuration as one of the following options:

FILEDSNPATH | FILEDSN_ROOT

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

PURGE_CACHE *time-out-value*

Forces the removal of outdated cache tables by specifying, in minutes, how often old data cache tables are removed from the server. A positive time value indicates the interval at which the cleanup thread checks for items to purge. A negative time-out-value means that cleanup happens only in response to an explicit PURGE CACHE command. A value of **0** indicates that old cache tables are removed after a **CREATE CACHE**, **REFRESH CACHE**, or **PURGE CACHE** command is issued.

SHUTDOWN_TIMEOUT
```
SHUTDOWN_TIMEOUT time-out-value
```

Specifies a time out (in seconds) for forcing a disconnect of sessions during a server shutdown event. If this value is set to **1**, the server should exit immediately. Specifying a **0**, a negative value, or dropping the value specifies that the server will wait for normal shutdown. The default value is **0**, which allows a normal shutdown, where the server will wait for all current connections to complete and close before initiating shutdown.

TRACEFILEPATH | TRACEFILE_ROOT
```
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 can be absolute or relative to the server's ContentRoot that is specified in the server configuration file. The default trace file path is **C:\Program Files\SASHome \SASFederationServer\4.2**. The **DRIVER_TRACEFILE=** path that is set in the connection option, **DRIVER TRACE**, is resolved against the path that is set here.

SHARED LOGIN KEY
```
SHAREDLOGINKEY key
```

The SAS Metadata Server grouping key used to search for shared login map credentials.

PL (Procedure Language) SOURCE MANAGEMENT SECURITY
```
PLSrcMgtSecurity TRUE | FALSE
```

Specifies procedure language (PL) source management authorization, which allows DS2 users to publish content to a schema specified in the connection. The default is TRUE, which limits DS2 content publishing to administrators and owners of the schema containing code. When granular PL security is set to FALSE, regular DS2 users may publish or withdraw code without being a SAS Federation Server administrator, or owner of the schema containing the code package. To allow this access, you must set PL source management authorizations to FALSE, as shown in the following example: **ALTER SERVER {OPTIONS XSET PLSrcMgtSecurity FALSE}**. If this option is explicitly set and later dropped, it reverts to the default of TRUE.

PACKAGE (Data Masking) ENCRYPT_KEY, RANDOM_SEED
Data masking encryption is configured as a PACKAGE option using ENCRYPT_KEY or RANDOM_SEED with specific parameters. The **SYSCAT.DM.MASK** function accepts defaults configured as package options in addition to the various arguments associated with each rule type, for example, the KEY argument for the ENCRYPT rule type defaults to the value configured as the **ENCRYPT_KEY** package option.

The following example sets a default encryption key for use with ENCRYPT and HASH:

```
ALTER SERVER {options PACKAGE(name 'DM',
      ENCRYPT_KEY '212e8ba6b7f84796a87a985d54277f2f')}
```

The following example drops the ENCRYPT_KEY option:

```
alter server {options PACKAGE(name DM drop ENCRYPT_KEY)}
```

RANDOM_SEED specifies an integer that is used with the RANDOM, RANDIG, RANSTR, and RANDATE data masking functions.

```
alter server {OPTIONS  package (name DM, set RANDOM_SEED 98765)}
```

The following example drops the RANDOM_SEED option:

```
alter server {options PACKAGE(name DM, drop RANDOM_SEED)}
```

CONNECTION POOLING

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

This option controls connection pooling for the server. If connection pooling is enabled, database connections are not immediately disconnected upon client request. Instead, the connections are put into a connection pool so that they can be reused by subsequent connection requests for the same database with the same attributes and credentials.

CONNECTION POOLING 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 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 POOLING MAXSIZE

```
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 not set 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 a list of cache options and can be one of the following:

```
CACHE ( NAME cache-name, cache-property cache-property value )
```

The **NAME** option specifies the name of the cache. 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 service (AS) 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.

cache-property : : =

Used to specify a property of the cache.

    TIMEOUT

        TIMEOUT=*n*

        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. **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 not set through a parent namespace.

CONOPTS

Use the **CONOPTS** connection string option to call the FedSQL driver and set SQL statement limits.

FedSQL Driver

    CONOPTS(driver FEDSQL)

FedSQL is the required driver for SQL requests on SAS Federation Server.

SQL Statement Limit

    DEFAULT_ATTR(SQL_STMT_MEM_LIMIT = *n*)

Use the **SQL_STATEMENT_LIMIT** connection string option to control the amount of memory available to answer SQL requests, and enforce this limit for all connections. If the option is specifically set on a particular DSN, then the DSN value should override the system setting. (*n*)umber is treated as an integer and is specified in bytes.

Examples

These examples set a trace file path:

```
ALTER SERVER {OPTIONS add TRACEFILEPATH "C:\tracefiles"}
```

```
ALTER SERVER {OPTIONS add TRACEFILEPATH "logs\tracefiles"}
```

Set the shutdown time-out so the server will wait two minutes before disconnecting during a shutdown event:

```
ALTER SERVER {options (xset SHUTDOWN_TIMEOUT 120) }
```

Add a shared login key:

```
ALTER SERVER {OPTIONS xset SHAREDLOGINKEY 'DefaultKey'}
```

Set authentication service cache options:

```
ALTER SERVER {OPTIONS( CACHE(NAME AS.Subject, TIMEOUT 300),
```

```
        CACHE(NAME AS.List.Subjects, TIMEOUT 60) )}

    ALTER SERVER {OPTIONS(xset PURGE_CACHE 30)}
```

Add connection options for the FedSQL driver:

```
    ALTER SERVER {options CONOPTS(driver FEDSQL,
    xset DEFAULT_ATTR(SQL_STMT_MEM_LIMIT 8000000))}
```

This statement sets an encryption key, or random seed, for data masking.

```
    ALTER SERVER {OPTIONS  package (name DM, set RANDOM_SEED 98765)}
```

Additional data masking examples are shown above in the PACKAGE server option.

Data Services

CREATE DATA SERVICE

Description

The **CREATE [DATA] SERVICE** statement enables you to create a data service. When you create a data service, a DSN with the same name is also created.

Parameters

```
CREATE [DATA] SERVICE data-service-name

 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-name*
Specifies the name of the data service. The following rules apply:

- The name specified cannot match the reserved name of the internal data service, BASE.

- The name specified cannot match the name of an existing data service.

- The name specified cannot 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. Here is a list of valid data service types:

```
DB2UNXPC
FEDSVR
GENERIC
GENERIC_FED
GREENPLUM
HIVE
MDS
NETEZZA
ODBC
```

```
ODBC_FED
ORACLE
POSTGRES
SAP
SAPHANA
SASHDAT
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 SAS Metadata Server domain name associated with the specified data service. If omitted, the default domain name matches the data service name. The domain is checked against the authentication server and if not valid, will return an error only if VALIDATE is specified for the data service. If VALIDATE is not specified, a warning is presented but the data service is still created.

REGISTER [(*catalog-name1* **[,** *catalog-name2***...])** | *ALL*]

REGISTER is an optional catalog registration specification. Specify a list of catalogs to register or use the **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 **"userid"** requires double quotation marks and the **'password'** requires single quotation marks:

```
    UID "userid"

    PWD 'password'
```

The data service user ID and password are used for making the connection to the data source so that the connection and catalog names can be validated.

VALIDATE specifies whether the connection, domain and catalog names are to be validated:

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

If UID or PWD is specified, but VALIDATE is not, VALIDATE runs with a default of TRUE. If a UID or PWD is specified and VALIDATE is specified without a value, it defaults to TRUE. If VALIDATE is specified as TRUE but UID or PWD are not specified, then personal credentials are extracted on behalf of the caller, from the domain associated with the data service. The statement returns **TKTS_ERROR** if credentials do not exist in the domain.

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.

> **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 format is **CONOPTS( driver ODBC, CONOPTS( ... ) )**. The inner **CONOPTS** option, within the parenthesis, is a list-valued driver connection string, while the outer **CONOPTS** option groups arbitrary driver connection string options configured for the service.

>> *driver-connection-string-option*
>>> Specifies the connection options that correspond to the driver 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 the associated 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 are used. If not specified, the value for case sensitivity is set to the default for the data service. The defaults are **TRUE** (sensitive) for DB2, Oracle, and GreenPlum, and **FALSE** (insensitive) for all other data 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

This example creates an Oracle data service with connection options that specify a client driver and path.

```
CREATE SERVICE ORASERV TYPE ORACLE domain ORA1 {OPTIONS ( conopts
( Driver odbc, conopts(DSN tktsora)), conopt (Driver oracle, PATH tktsora) ) }
```

Create a DB2 data service that points to a DEV1 database:

```
CREATE SERVICE DB2_SERVICE TYPE DB2UNXPC domain DB2 {OPTIONS
 ( conopts (DB DEV1) ) }
```

Create a Teradata data service with a connection option for a server:

```
CREATE SERVICE TERA_SERVICE TYPE TERADATA domain TERA {OPTIONS
 ( conopts (Server kaching.unx.df.com) ) }
```

Create an SAP data service with connection options defining host, SAP system number and active batch job processing:

```
CREATE SERVICE SAPSERV TYPE SAP DOMAIN SAPDOMAIN {OPTIONS
 conopts(ashost apsrv.sup.com, sysnr 03, batch 1)}
```

ALTER DATA SERVICE

Description
Use ALTER DATA SERVICE to change the name or options of an existing data service.

Parameters

```
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 userID*
> 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 an argument or a Boolean value is not specified. VALIDATE also defaults to true if a UID and PWD pair are not 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 credentials do not 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 format is **CONOPTS(driver ODBC, CONOPTS(...))**. The inner **CONOPTS** option, within the parenthesis, is a list-valued driver connection string, while the outer **CONOPTS** option 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 **DRIVERdriver-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. For a list of valid connection string options, see the driver reference topic for your specific 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

This example is altering the ORACLE3 data service to define driver and DSN connection options:

```
ALTER DATA SERVICE ORACLE3 {OPTIONS conopts(Driver odbc,
 ODBC_DSN tktsora)}
```

Rename the ORACLE3 data service to ORACLE3_VER2:

```
ALTER DATA SERVICE ORACLE3 RENAME TO ORACLE3_VER2
```

This example drops the ODBC driver from the ORACLE3_VER2 data service:

```
ALTER DATA SERVICE ORACLE3_VER2 {OPTIONS DROP conopts(driver odbc) }
```

Define a catalog for the SERVICE1 data service:

```
ALTER SERVICE SERVICE1 CATALOG NEWCATALOG
```

DROP DATA SERVICE

Description
Use DROP DATA SERVICE to drop a specific data service.

Parameters
```
DROP [DATA] SERVICE data-service-name [drop-disposition]
```

data-service-name
> Specifies the name of the data service to drop.

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

This example drops the ORACLE3 data service:

```
drop DATA SERVICE ORACLE3
```

This example drops the MYSQL_SERVICE data service and all objects associated with it:

```
drop service "MYSQL_SERVICE" cascade force
```

This example drops the ORACLE1 data service and suppresses error messages:

```
drop data service ORACLE1 cascade
```

Data Source Names (DSN)

CREATE DSN

Description

Use the CREATE DSN statement to create a standard, single-source DSN, or a federated DSN. A federated DSN is a grouping of one or more standard DSNs. The Federation Server driver connection string options are an extension of the ODBC syntax. The connection string is a series of keyword/value pairs that are separated by semicolons. The equal sign (=) connects each keyword with its value.

Parameters

```
CREATE DSN dsn-name

UNDER data-service

[DESCRIPTION 'description text']
```

Note: Avoid using the backslash character when specifying **DESCRIPTION**, except to represent a literal backslash that should be escaped to parse correctly.

```
[CONNECT 'driver-connection-string-options']

[create-dsn-options]

[AS ADMINISTRATOR]
```

Here is the syntax to create a federated DSN:

```
CREATE DSN dsn-name

[DESCRIPTION 'description text'] [create-dsn-options]
```

```
ADD "(" dsn-name ["," ...] ")"
[AS ADMINISTRATOR]
```

Note: Avoid using the backslash character when specifying **DESCRIPTION**, except to represent a literal backslash that should be escaped to parse correctly.

```
[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. This option applies only to a standard DSN and is required.

```
UNDER data-service-name
```

The following naming 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']
```

Note: Avoid using the backslash character when specifying **DESCRIPTION**, except to represent a literal backslash where the string should be escaped to parse correctly.

CONNECT | NOPROMPT *driver-connection-string-options*
Specifies connection string options for standard DSNs only. Use NOPROMPT to suppress dialog box messages.

```
[CONNECT | NOPROMPT driver-connection-string-options']
```

The connection string options for the Federation Server drivers are an extension of the ODBC syntax that specifies options as semicolon-delimited **key=value** pairs. For more information about connection options and advanced options specific to your data source, see the Driver Reference chapter.

create-dsn-options
Specifies what options are included with the DSN.

dsn-config-options
Specifies the options to configure with 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:

LANG FEDSQL | DS2 | NO

Specifies whether to use FedSQL, DS2 or native dialect. The dialect defaults to FedSQL for BASE DSNs and all secured DSNs. The LANG (FEDSQL) option applies to both standard and federated DSNs. Note that you can execute only the language specified in the LANG setting. FedSQL and DS2 cannot be used together.

```
LANG=FEDSQL, LANG=DS2, LANG=NO
```

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 SAS 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*} ...] ")"

```
cso-value ::= PERSONAL |SHARED
```

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)**. Credentials Search Order applies only to standard DSNs.

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.

AS ADMINISTRATOR

```
[ 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.

Examples

Here are examples of the CREATE DSN statement:

```
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 CONNECT 'DATA_SERVICE=BASE;
LOCKTABLE=EXCLUSIVE'
```

```
CREATE DSN BASEDSN under BASE CONNECT 'CATALOG="catalog1_BASE";
LOCKTABLE=EXCLUSIVE'
```

```
CREATE DSN BASEDSN under BASE CONNECT '(CATALOG="catalog1_BASE";
LOCKTABLE=SHARE);(CATALOG="catalog2_BASE";LOCKTABLE=EXCLUSIVE)'
```

```
CREATE DSN BASEDSN under BASE CONNECT '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 "DB2Users" UNDER "Oracle Service" CONNECT 'DRIVER=Oracle;GROUP=DB2Users'
 {OPTIONS CSO PERSONAL} AS ADMINISTRATOR
```

```
CREATE DSN MYFEDERATED_DSN ADD (mydsn1, mydsn2, mydsn3)
AS ADMINISTRATOR
```

ALTER DSN

Description

Use the ALTER DSN statement to change the name of a standard or federated DSN, or alter advanced options. Advanced options are specific to the data source that the DSN applies to. For information about the advanced options, see the Driver Reference topic for your data source.

Parameters

Here is the syntax to alter a standard DSN:

```
ALTER DSN dsn-name alter-dsn-options
```

```
ALTER DSN dsn-name RENAME TO new-dsn-name
```

Here is the syntax to alter a federated DSN:

```
ALTER DSN dsn-name ADD "(" dsn-name ["," ...] ")"
```

```
ALTER DSN dsn-name DROP "(" dsn-name ["," ...] ")"
```

Note: The ADD and DROP options are valid with a federated DSN only.

dsn-name
Specifies the DSN name.

alter-dsn-options
Specifies the options to alter.

```
alter-dsn-options

    ::= create-dsn-options
```

new-dsn-name
Specifies the new DSN 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']
```

Note: Avoid using the backslash character when specifying **DESCRIPTION**, except to represent a literal backslash where the string should be escaped to parse correctly.

CONNECT | NOPROMPT *driver-connection-string-options*
Specifies connection string options for standard DSNs only. Use NOPROMPT to suppress dialog box messages.

```
[CONNECT | NOPROMPT driver-connection-string-options']
```

The connection string options for the Federation Server drivers are an extension of the ODBC syntax that specifies options as semicolon-delimited **key=value** pairs. For more information about connection options and advanced options specific to your data source, see the Driver Reference chapter.

Examples

Here are examples of the ALTER DSN statement:

```
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}
```

DROP DSN

Description

Use the DROP DSN statement to drop a server-based DSN.

Parameters

```
DROP DSN dsn-name [FORCE]
```

dsn-name
> Specifies the name of the DSN.

FORCE Use FORCE to suppress error messages when the DSN does not exist.

Examples

Here are examples of the DROP DSN statement:

```
DROP DSN "DSN1"

DROP DSN "DSN1" FORCE
```

Catalogs and Schemas

CREATE CATALOG

Description

Use CREATE CATALOG to create a catalog under a specific data service.

`TIP` Use double quotation marks when specifying a catalog name.

Parameters

```
CREATE CATALOG "catalog" UNDER data-service

  [ NATIVE NAME native-name ]

  [ create-catalog-options ]
```

"catalog"
> Specifies a logical catalog name. Surround the catalog name in double quotation marks.

data-service
> Specifies the data service name under which the catalog is to be created.

native-name
> Specifies the native catalog name if the native catalog name is not unique within the server. The native name is used resolve catalog name collisions between multiple data services that support catalogs. Client SQL always references the catalog through the logical catalog name regardless of whether a native name is specified. The native catalog is treated as a missing value if it reflects the same name as the logical catalog.

create-catalog-options
> Specifies the options to create a catalog. This option applies only to the BASE data service.

```
create-catalog-options ::=
create-catalog-options::=
conopts-configuration-list
```

conopts-configuration-list
> If **DRIVERdriver-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 specified driver. For a list
> of valid connection string options, see the driver reference topic for your specific
> data source.

Examples

These examples create a catalog under a specific data service:

```
CREATE CATALOG "catalog1_BASE" UNDER BASE
```

```
CREATE CATALOG "TKTEST" UNDER SQLServer1
```

This example creates a logical catalog (CATALOG1) and a native catalog (TKTEST)
under the SQLSERVER1 data service:

```
CREATE CATALOG "Catalog1" UNDER SQLServer1 NATIVE NAME "TKTEST"
```

Catalog C1 created under a BASE data service with the COMPRESS connection option
that activates row compression for SAS data sets:

```
CREATE CATALOG "c1" UNDER BASE {OPTIONS conopts (COMPRESS YES)}
```

ALTER CATALOG

Description
Use ALTER CATALOG to change the name of a catalog, rename a native catalog, or
alter options.

Parameters
```
ALTER CATALOG "catalog" RENAME TO "newcatalogname"

ALTER CATALOG "catalog"

    [ NATIVE NAME "native-name" ]

    [ alter-catalog-options ]
```

"catalog"
> Specifies the name of the existing catalog.

"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 applies only 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
Here are examples of the ALTER CATALOG statement:

```
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)}
```

DROP CATALOG

Description
Use the DROP CATALOG statement to drop a catalog.

Parameters
```
DROP CATALOG "catalog" [ drop-disposition ]
```

"*catalog*"
Specifies the catalog name. Catalog name requires double quotation marks.

drop-disposition
```
drop-disposition ::=
{RESTRICT | CASCADE} [FORCE]
```

Specifies a drop disposition as one of the following values:

RESTRICT   Specifies that the drop target is empty. This is the default value.

CASCADE   Specifies that container 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
This example drops "CATALOG3":

```
DROP CATALOG "catalog3"
```

This example drops "CATALOG1_BASE" and all associated container objects:

```
DROP CATALOG "catalog1_base" cascade
```

CREATE SCHEMA

Description
Use the CREATE SCHEMA statement to create a schema and designate an owner.

Parameters
```
CREATE SCHEMA [ "catalog"."schema"]

    [ AUTHORIZATION|OWNER owner ]

    [ create-schema-options ]
```

"catalog"
> Specifies the optional catalog name under which to create the schema. This is useful for data sources that support catalog names. For those that do not, the catalog name must be the logical catalog name which defaults to the name of the data service.

"schema"
> Specifies the name of the schema.

owner
> Authorization identifier of the schema owner. If the **AUTHORIZATION** clause is not specified, schema ownership defaults to the **SYSTEM** user account. However, schema ownership by the **SYSTEM** user account could present problems with FedSQL views and data cache.

create-schema-options
> Specifies an option list for the schema.

> ```
> create-schema-options ::=
>
> "{" OPTIONS ["("] schema-option
>
> [ { "," schema-option } ... ] [")"] "}"
> ```

> *schema-option*
>> Specifies the syntax for schema options. This option applies only 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 **DRIVERdriver-name**. For a list of valid connection string options, see the driver reference topic for your specific data source.

>> **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 is required for BASE schemas.

>>> ```
>>> PRIMARYPATH path ::=
>>>
>>>   quoted-identifier
>>> ```

>> *quoted-identifier*
>>> Specifies a single quoted or double quoted name.

Examples
Here are examples of the CREATE SCHEMA statement:

```
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)}
```

ALTER SCHEMA

Description
Use the ALTER SCHEMA statement to change the name or options of an existing schema. You can also alter advanced options that are driver–specific.

Parameters
```
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 to change.

"newschema"
> Specifies the new schema name.

alter-schema-options
> Specifies what options to alter in the schema.
>
> ```
> alter-schema-options ::=
>
>   "{" OPTIONS ["("] alter-schema-option
>
> [{"," alter-schema-option} ... ] [")"]"}"
> ```
>
> *alter-schema-option*
>> Specifies the schema option to alter. This option applies only 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 **DRIVERdriver-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 **DRIVERdriver-name**. For a list of valid connection string options, see the driver reference topic for your specific data source.

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

Here are examples of the ALTER SCHEMA statement:

```
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)}
```

DROP SCHEMA

Description

Use DROP SCHEMA to drop a schema.

Parameters

```
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

Here are examples of the DROP SCHEMA statement:

```
DROP SCHEMA "catalog1_BASE"."schema1_BASE"

DROP SCHEMA "catalog1_BASE"."schema1_BASE" force
```

Cache

CREATE CACHE

Description

Use the CREATE CACHE statement to create a cache definition for a specific view. You can also specify options for the cache.

Parameters

```
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. Issue a separate REFRESH CACHE command to create and populate the cache table.

FORCE

The cache table and definition is retained even if an error occurs during REFRESH.

EXEC | EXECUTE

Use the EXEC | EXECUTE command with the BEFORE, AFTER, and CLEANUP options as specified below.

BEFORE

```
BEFORE (fedsql_syntax) [FORCE]
```

The statements in fedsql_syntax will be executed before the cache table is created and populated. The FORCE option suppresses any errors.

AFTER

```
AFTER (fedsql_syntax) [FORCE]
```

The statements in fedsql_syntax will be executed after the cache table is created and populated. The FORCE option suppresses any errors.

CLEANUP

```
CLEANUP (fedsql_syntax) [FORCE]
```

The statements in fedsql_syntax will be executed only in the event of an error during creation or population of the cache table. The FORCE option will suppress any errors.

Use the following keywords with the USING, BEFORE, AFTER, and CLEANUP clauses. The keywords must be in UPPERCASE and contain no blank spaces with the brackets.

{CACHE}

Expands to a fully qualified cache table name using "double quotation marks".

{CACHE_CATALOG}

Expands to a cache catalog name. Does not use quotation marks.

{CACHE_SCHEMA}

Expands to a cache schema name. Does not use quotation marks.

{CACHE_TABLE}

Expands to a cache table name. Does not use quotation marks.

SET

```
SET    option_name=value
```

Specifies additional options and values to use during cache creation and population. The options are listed below.

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. Valid options are **FORCE|FORCE_COL|FORCE_COL_SIZE|STRICT|SAFE**.

RESTART

```
RESTART='REFRESH'
```

REFRESH sets active or deferred caches to automatically refresh after each server start-up. Suspended or disabled caches are not affected.

Note: FedSQL requires that option values be enclosed in single quotation marks. Any other value for RESTART= will produce an error. This value is case sensitive.

Examples

Here is an example of the CREATE CACHE statement:

```
CREATE CACHE ORACLE_SERVICE.TKTSTST1.view_red in "SAMPLE"."tktstst1"
    set CT_PRESERVE='SAFE'
```

ALTER CACHE

Description

With the ALTER CACHE statement, you can disable, enable or refresh cache tables. ALTER CACHE requires the CREATE CACHE or ALTER CACHE privilege.

Parameters

```
ALTER CACHE "catalog"."schema"."view" OPTION
```

"catalog"."schema"."view"

Specifies the catalog, schema and name of the view.

OPTION

Specifies an option for the statement as one of the following:

DISABLE

Disables use of the specified cache. References to the view use the view rather than the cached data.

ENABLE

Enables use of the specified cache after having been disabled.

REFRESH
> Refreshes the cache table for the specified view.

Examples

Here are examples of the ALTER CACHE statement:

```
ALTER CACHE "catalog1"."schema1"."view1" DISABLE

ALTER CACHE [ "catalog2"."schema2". ] view2 ENABLE

ALTER CACHE [ "catalog3"."schema3". ] view3 REFRESH
```

DROP CACHE

Description

Use DROP CACHE to drop a cached view.

Parameters

```
DROP CACHE [ "catalog"."schema". ] view [FORCE ]
```

"catalog"."schema"
> Specifies the catalog and schema of the data cache view.

view
> Specifies the name of the data cache view.

> FORCE   Suppresses error messages if the cache to be dropped does not exist.

.

PURGE CACHE

Description

PURGE CACHE forces the removal of cache tables that are no longer needed. The ability to purge a cache is limited to system users, administrators, and users who have CREATE CACHE permission set for a server object.

Note: The PURGE CACHE statement is not the same as the PURGE CACHE option in the ALTER SERVER statement. Only system users or administrators (those with ADMINISTER privilege) on the SAS Federation Server can execute the PURGE CACHE option with the ALTER SERVER statement.

Parameters

```
PURGE CACHE
```

Appendix 2

Information Views

About Information Views

Overview

Information views are a summary of configuration and security activities for SAS Federation Server that are stored as metadata in the system catalog (SYSCAT) . You can query any view using regular SELECT statements, for example, if you need to know something about data structure or privileges for a specific object. However, you must have privileges to view the data. The visibility rules that apply are outlined below.

Visibility Rules

User Privileges

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. There are 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.

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	A catalog is visible to a user if: • the data service is visible.
SCHEMAS CONFIG_SCHEMAS	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 SASUSERS 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 SASUSERS group. • the privilege is granted in the PUBLIC group.

AUTHORIZATION_IDENTIFIERS

The AUTHORIZATION_IDENTIFIERS view displays SAS Metadata 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 SAS Metadata Server. Removing an object from the SAS Metadata Server does not remove it from the AUTHORIZATION_IDENTIFIERS view. Use the 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 SAS Metadata Server. NULL is displayed if the group, role, or user is orphaned.
ID	VARCHAR(256)	Specifies the authorization identifier from the SAS Metadata Server.
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

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.

Name	Type	Description
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" on page 279.
NUM_ROWS	BIGINT Nullable	Specifies the number of rows inserted into the data cache table. Note that this should be considered a rough estimate.
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.

Here are the status rules regarding 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

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.

Name	Type	Description
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

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).
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.

Name	Type	Description
GRANTABLE	CHAR(1)	Specifies if the privilege can be granted to others. The only valid value is N (No).

COLUMNS

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

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).

Name	Type	Description
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

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

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.

Name	Type	Description
DATA_SERVICE_NAME	VARCHAR(256)	Specifies the unique name of the data service.
SEQUENCE	SMALLINT	Specifies the configuration chunk sequence.

CONFIG_DSNS

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

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, such as **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

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.

Name	Type	Description
SEQUENCE	SMALLINT	Specifies the configuration chunk sequence.

DATA_SERVICES

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.
TYPE	CHAR(32)	Keyword for the data type. Valid values include: BASE DB2 GREENPLUM HIVE MDS ODBC ODBC_FED ORACLE TRAN TERADATA SAP SQLSVR Other values are possible.
DOMAIN	VARCHAR(256)	The SAS Metadata Server domain that is associated with the data service.

DATA_SOURCE_NAMES

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.
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

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.

Name	Type	Description
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).
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

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

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

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 whether 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

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 SAS Metadata 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

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 segments that will fit within the DATA column as it is defined. The SEQUENCE column orders the entries within a MESSAGE_ID and MESSAGE_NUM identifies the individual messages.

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

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 – FedSQL View 3 – Cache 4 – Procedure/Function 5 – Any relation 6 – Not used 7 – Package 8 – DS2 Thread
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

The OBJECT_PRIVILEGES view displays the privileges for SAS Federation Server objects. 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 in the OBJECT_PRIVILEGES view:

Name	Type	Description
CATALOG_NAME	VARCHAR(256)	Specifies the name of the catalog.

Name	Type	Description
SCHEMA_NAME	VARCHAR(256)	Specifies the name of the schema.
OBJECT_NAME	VARCHAR(256)	Specifies the name of the object.
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

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 or the user's group membership.

The PRIVILEGES view contains rows for users and groups that have any direct privileges 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 view contains rows for all users and groups that have any direct privileges set, or privileges inherited as a result of 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 direct privileges.

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 in 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.
GRANTEE_TYPE	CHAR(1)	Specifies the grantee type as U (User) or G (Group).

Name	Type	Description
PRIVILEGE_NAME	VARCHAR(20)	Specifies the privilege name as one of the following values: SELECT UPDATE EXECUTE 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 whether 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 - 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

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

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.
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

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).

Name	Type	Description
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.
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

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 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.

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).

Name	Type	Description
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).
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

Appendix 3

Legal Notices

Apache Portable Runtime License Disclosure

Copyright © 2008 DataFlux Corporation LLC, Cary, NC USA.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Apache/Xerces Copyright Disclosure

The Apache Software License, Version 3.1

Copyright © 1999-2003 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE

FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., http://www.ibm.com. For more information on the Apache Software Foundation, please see http://www.apache.org.

Boost Software License Disclosure

Boost Software License - Version 1.0 - August 17, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Canada Post Copyright Disclosure

The Data for areas of Canada includes information taken with permission from Canadian authorities, including: © Her Majesty the Queen in Right of Canada, © Queen's Printer for Ontario, © Canada Post Corporation, GeoBase®, © Department of Natural Resources Canada. All rights reserved.

DataDirect Copyright Disclosure

Portions of this software are copyrighted by DataDirect Technologies Corp., 1991 - 2008.

Expat Copyright Disclosure

Part of the software embedded in this product is Expat software.

gSOAP Copyright Disclosure

IBM Copyright Disclosure

OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE
FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL
DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF
USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in
advertising or otherwise to promote the sale, use or other dealings in this Software
without prior written authorization of the copyright holder.

Informatica Address Doctor Copyright Disclosure

AddressDoctor® Software, © 1994-2015 Platon Data Technology GmbH

Loqate Copyright Disclosure

The Customer hereby acknowledges the following Copyright notices may apply to
reference data.

Australia: Copyright. Based on data provided under license from PSMA Australia
Limited (www.psma.corn.au)

Austria: © Bundesamt für Eich- und Vermessungswesen

Brazil: Conteudo firnecido por MapLink. Brazil POIs may not be used in publically
accessible, internet-based web sites whereby consumers obtain POI data for their
personal use.

Canada:

Copyright Notice: This data includes information taken with permission from Canadian
authorities, including © Her Majesty, © Queen's Printer for Ontario, © Canada Post,
GeoBase ®.

End User Terms: The Data may include or reflect data of licensors including Her
Majesty and Canada Post. Such data is licensed on an "as is" basis. The licensors,
including Her Majesty and Canada Post, make no guarantees, representation, or
warranties respecting such data, either express or implied, arising by law or otherwise,
including but not limited to, effectiveness, completeness, accuracy, or fitness for a
purpose. The licensors, including Her Majesty and Canada Post, shall not be liable in
respect of any claim, demand or action, irrespective of the nature of the cause of the
claim, demand or action alleging any loss, injury or damages, direct or indirect, which
may result from the use or possession of the data or the Data.

The licensors, including Her Majesty and Canada Post, shall not be liable in any way for
loss of revenues or contracts, or any other consequential loss of any kind resulting from
any defect in the data or in the Data.

End User shall indemnify and save harmless the licensors, including Her Majesty the
Queen, the Minister of Natural Resources of Canada and Canada Post, and their officers,
employees and agents from and against any claim, demand or action, irrespective of the
nature of the cause of the claim, demand or action, alleging loss, costs, expenses,
damages, or injuries (including injuries resulting in death) arising out of the use of
possession of the data or the Data.

Croatia, Cyprus, Estonia, Latvia, Lithuania, Moldova, Poland, Slovenia, and/or Ukraine:
© EuroGeographics

France: source: Géoroute® IGN France & BD Carto® IGN France

Germany: Die Grundlagendaten wurden mit Genehmigung der zuständigen Behörden entnommen

Great Britain: Based upon Crown Copyright material.

Greece: Copyright Geomatics Ltd. Hungary: Copyright © 2003; Top-Map Ltd.

Italy: La Banca Dati Italiana è stata prodotta usando quale riferimento anche cartografia numerica ed al tratto prodotta e fornita dalla Regione Toscana.

Norway: Copyright © 2000; Norwegian Mapping Authority

Portugal: Source: IgeoE – Portugal

Spain: Información geográfica propiedad del CNIG

Sweden: Based upon electronic data © National Land Survey Sweden.

Switzerland: Topografische Grundlage © Bundesamt für Landestopographie.

Microsoft Copyright Disclosure

Microsoft®, Windows, NT, SQL Server, and Access, are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle Copyright Disclosure

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates.

PCRE Copyright Disclosure

A modified version of the open source software PCRE library package, written by Philip Hazel and copyrighted by the University of Cambridge, England, has been used by DataFlux for regular expression support. More information on this library can be found at: ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

Copyright © 1997-2005 University of Cambridge. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

• Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

• Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Red Hat Copyright Disclosure

Red Hat® Enterprise Linux®, and Red Hat Fedora™ are registered trademarks of Red Hat, Inc. in the United States and other countries.

SAS Copyright Disclosure

Portions of this software and documentation are copyrighted by SAS® Institute Inc., Cary, NC, USA, 2009. All Rights Reserved.

SQLite Copyright Disclosure

The original author of SQLite has dedicated the code to the public domain. Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

Sun Microsystems Copyright Disclosure

Java™ is a trademark of Sun Microsystems, Inc. in the U.S. or other countries.

TomTom Copyright Disclosure

© 2006-2015 TomTom. All rights reserved. This material is proprietary and the subject of copyright protection, database right protection, and other intellectual property rights owned by TomTom or its suppliers. The use of this material is subject to the terms of a license agreement. Any unauthorized copying or disclosure of this material will lead to criminal and civil liabilities.

USPS Copyright Disclosure

National ZIP®, ZIP+4®, Delivery Point Barcode Information, DPV, RDI, and NCOALink®. © United States Postal Service 2005. ZIP Code® and ZIP+4® are registered trademarks of the U.S. Postal Service.

DataFlux is a non-exclusive interface distributor of the United States Postal Service and holds a non-exclusive license from the United States Postal Service to publish and sell USPS CASS, DPV, and RDI information. This information is confidential and proprietary to the United States Postal Service. The price of these products is neither established, controlled, or approved by the United States Postal Service.

VMware Copyright Disclosure

VMware® virtual environment provided those products faithfully replicate the native hardware and provided the native hardware is one supported in the applicable DataFlux

product documentation. All DataFlux technical support is provided under the terms of a written license agreement signed by the DataFlux customer.

The VMware virtual environment may affect certain functions in DataFlux products (for example, sizing and recommendations), and it may not be possible to fix all problems.

If DataFlux believes the virtualization layer is the root cause of an incident; the customer will be directed to contact the appropriate VMware support provider to resolve the VMware issue and DataFlux shall have no further obligation for the issue.

Recommended Reading

Here is the recommended reading list for SAS Federation Server:

- SAS Federation Server Manager: User's Guide

- SAS Drivers for Federation Server: User's Guide

- SAS FedSQL Language Reference

- SAS DS2 Language Reference

- SAS LIBNAME Engine for SAS Federation Server: User's Guide

- SAS Intelligence Platform: System Administration Guide (SAS Metadata Server)

- SAS Management Console: Guide to Users and Permissions

- SAS Guide to Metadata-Bound Libraries

For a complete list of SAS publications, go to sas.com/store/books. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: sas.com/store/books

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.