## Contents


### PART 1  Introduction  1

**Chapter 1 • Introduction to the Administrator's Guide**  ............................................................................ 3
- Overview of SAS In-Database Products ...................................................... 3
- What Is Covered in This Document? .......................................................... 4

### PART 2  Administrator's Guide for Aster  7

**Chapter 2 • Aster Administrator's Guide**  .................................................................................. 9
- Prerequisites .................................................................................. 9
- Overview of the SAS Embedded Process for Aster ........................................ 9
- Aster Permissions ........................................................................... 10
- Documentation for Using In-Database Processing in Aster ......................... 10

### PART 3  Administrator's Guide for DB2  11

**Chapter 3 • DB2 Administrator's Guide**  .............................................................................. 13
- Prerequisites .................................................................................. 13
- Overview of the In-Database Deployment Package for DB2 ....................... 14
- Function Publishing Process in DB2 ....................................................... 14
- DB2 Installation and Configuration ......................................................... 15
- Validating the Publishing of SAS_COMPILEUDF and
  SAS_DELETEUDF Functions and Global Variables ..................................... 25
- DB2SET Command Syntax for In-Database Processing ............................... 26
- Controlling the SAS Embedded Process for DB2 ....................................... 27
- DB2 Permissions ........................................................................... 28
- Documentation for Using In-Database Processing in DB2 ......................... 30

### PART 4  Administrator's Guide for Greenplum  31

**Chapter 4 • Greenplum Administrator's Guide**  ................................................................. 33
- Prerequisites .................................................................................. 33
- Overview of the In-Database Deployment Package for Greenplum ............... 34
- Greenplum Installation and Configuration ................................................ 35
- Validation of Publishing Functions ......................................................... 45
- Controlling the SAS Embedded Process ................................................... 46
- Semaphore Requirements When Using the SAS Embedded Process for Greenplum .............................................................................................................. 47
- Greenplum Permissions ....................................................................... 48
What’s New in SAS 9.4 In-Database Products: Administrator’s Guide

Overview

In SAS 9.4, the following new features and enhancements were added to expand the capabilities of the SAS In-Database products:

- In the November 2018 release of SAS 9.4, the following changes and enhancements were made:
  - In-database scoring for the SAS Scalable Performance Data Server is no longer supported.
  - Information about IBM BigInsights and Pivotal has been removed from the documentation. These Hadoop vendors have asked their customers to move to Hortonworks.
  - The information for deploying the SAS Embedded Process for all data sources has been removed from this document. The SAS Embedded Process deployment package is now being delivered from one location and the deployment is the same for both SAS 9.4 and SAS Viya. The documentation is now available in *SAS Embedded Process: Deployment Guide*.
    
    The installation information for the in-database deployment package containing the SAS Formats Library and binary files remains in this document. This package is available for DB2, Greenplum, Netezza, and Teradata.
  
    - The SAS Deployment Manager is no longer used to deploy the SAS Embedded Process for Hadoop. The SAS Embedded Process deployment package is now being delivered from a SAS Viya repository and the deployment process has changed. It is the same for both SAS 9.4 and SAS Viya. The documentation is now available in *SAS Embedded Process: Deployment Guide*.
    
    - Due to changes to the deployment of the SAS Embedded Process, install scripts for SAS Data Quality Accelerator for Teradata stored processes and the script to deploy SAS Contextual Analysis text analytics models to the Hadoop nodes are delivered in a different location.
  
- In the December 2017 release of SAS 9.4, new versions of the SAS Embedded Process for Hadoop and Teradata are available. The SAS Embedded Process for other data sources did not change.

- In the September 2017 release of SAS 9.4, the following changes and enhancements were made:
  
    - The SAS Deployment Manager can now be run if the cluster manager is enabled with Transport Layer Security (TLS). The SAS Deployment Manager validates if the Certificate of Authority (CA) exists for the host that you are trying to access.
If the CA does not exist, a warning message is issued and asks to run another SAS Deployment Manager task to add the CA.

Note: All discussion of TLS is also applicable to the predecessor protocol, Secure Sockets Layer (SSL).

- If you license SAS Data Loader for Hadoop, you can now deploy the SAS Embedded Process and the SAS Data Quality Accelerator, the SAS Data Loader for Hadoop Spark Engine, and the SAS Quality Knowledge Base individually, all together, or in any combination at one time.

- In the November 2016 release of SAS 9.4, the following changes and enhancements were made:
  - The installation and configuration of the SAS Embedded Process for Aster, DB2, Greenplum, Netezza, Oracle, and SAP has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.
  - If you use the SAS In-Database Code Accelerator for Hadoop, the SPD Engine SerDe can be used to access Hive tables.
  - Several new properties have been added to the SAS Embedded Process for Hadoop ep-config.xml file that enable you to adjust performance.
  - Deploying the SAS Embedded Process for Hadoop no longer requires root or sudo access.
  - Changes have been made to the sasep-admin.sh SAS Embedded Process for Hadoop deployment script.
  - SAS in-database processing in Teradata now supports single sign-on (SSO) authentication.
  - IBM BigInsights and Pivotal HD now support the Ambari server cluster manager. You can now use the SAS Deployment Manager to install the in-database deployment package for Hadoop on IBM BigInsights and Pivotal HD.

- In the January 2016 release of SAS 9.4, the following changes and enhancements were made:
  - The removal of the SAS Embedded Process stack using Ambari has been simplified. The delete_stack.sh file now enables you to remove the ep-config.xml file, a specific version of the SAS Embedded Process, or all versions of the SAS Embedded Process.
  - A new panel that lists the products being installed by the SAS Deployment Manager has been added. Information that appears on other panels also reflects more closely what is being installed.

- In the July 2015 release of SAS 9.4, the following changes and enhancements were made:
  - The installation and configuration of the SAS Embedded Process for Teradata has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.
  - The installation and configuration of the SAS Embedded Process for Hadoop has changed significantly. For Cloudera and Hortonworks, Cloudera Manager and Ambari are used to install the SAS Embedded Process and the SAS Hadoop MapReduce JAR files. For IBM BigInsights, MapR, and Pivotal HD, the in-database deployment package is delivered to the client from the SAS Install Depot. In addition, the SAS Embedded Process and the SAS Hadoop MapReduce
JAR files are installed with one script instead of in two separate scripts. The new process has a smaller client footprint and is a faster install process.

- In the August 2014 release of SAS 9.4, the following changes and enhancements were made:
  - Numerous changes were made to the installation and configuration script for the SAS Embedded Process for Hadoop.

- In the April 2014 release of SAS 9.4, documentation enhancements were made in the following areas:
  - Additional information about the installation and configuration of the SAS Embedded Process for Hadoop was added.
  - Added semaphore requirements when using the SAS Embedded Process for Greenplum.

- In the December 2013 release of SAS 9.4, the following changes and enhancements were made:
  - New Hadoop JAR files are now tied to the version of Apache Hadoop that you are using.

- In the June 2013 release of SAS 9.4, the following changes and enhancements were made:
  - In-database scoring for Netezza has been enhanced by the addition of the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Netezza to read and write data.
  - The Hadoop scripts that install, control, and provide a status of the SAS Embedded Process have changed. There is now just one script, sasep-server.sh, that installs both the SAS Embedded Process and the Hadoop JAR files.

---

**SAS In-Database Code Accelerator**

**November 2016 Release of SAS 9.4: Changes and Enhancements**

If you use the SAS In-Database Code Accelerator for Hadoop, the SPD Engine SerDe can be used to access Hive tables.

**SAS 9.4: Changes and Enhancements**

The SAS In-Database Code Accelerator must be licensed at your site.
Aster Changes

November 2016 Release of SAS 9.4: Changes and Enhancements

The installation and configuration of the SAS Embedded Process for Aster has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.

DB2 Changes

November 2016 Release of SAS 9.4: Changes and Enhancements

The installation and configuration of the SAS Embedded Process for DB2 has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.

Greenplum Changes

November 2016 Release of SAS 9.4: Changes and Enhancements

The installation and configuration of the SAS Embedded Process for Greenplum has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.

April 2014 Release of SAS 9.4: Changes and Enhancements

Information about semaphore requirements when using the SAS Embedded Process was added to *SAS In-Database Products: Administrator’s Guide*.

SAS 9.4: Changes and Enhancements

There are several changes for Greenplum:

- Version 1.2 of the Greenplum Partner Connector (GPPC) is now available and should be installed if you use SAS Embedded Process 9.4.

- A new script, UninstallSASEPFiles.sh, is available. This script stops and uninstalls the SAS Embedded Process on each database host node.
Hadoop Changes

**November 2018 Release of SAS 9.4: Changes and Enhancements**

Information about IBM BigInsights and Pivotal has been removed from the documentation. These Hadoop vendors have asked their customers to move to Hortonworks.

**November 2016 Release of SAS 9.4: Changes and Enhancements**

In the November 2016 release of SAS 9.4, the following changes and enhancements were made:

- The number of SAS Embedded Process JAR files that are installed during deployment has been reduced from six to one.
- Several new properties have been added to the SAS Embedded Process ep-config.xml file that enable you to adjust performance.
- If you license SAS Data Loader for Hadoop, SAS Data Quality Accelerator for Teradata, or SAS Contextual Analysis In-Database Scoring for Hadoop, the Data Quality Accelerator for Hadoop, the Data Quality Accelerator for Teradata, and the SAS Contextual Analysis In-Database Scoring for Hadoop components, respectively, are included in the in-database deployment package for Hadoop and Teradata.
- Deploying the SAS Embedded Process for Hadoop no longer requires root or sudo access.
- IBM BigInsights and Pivotal HD now support the Ambari server cluster manager. You can now use the SAS Deployment Manager to install the in-database deployment package for Hadoop on IBM BigInsights and Pivotal HD.
- The sasep-admin.sh script has been changed:
  - A new option, -x, that enables you to deploy the SAS Embedded Process for Hadoop without root or sudo access has been added.
  - The -genconfig and -log options have been deleted.

**January 2016 Release of SAS 9.4: Changes and Enhancements**

In the January 2016 release of SAS 9.4, the following changes and enhancements were made:

- The removal of the SAS Embedded Process stack using Ambari has been simplified. The delete_stack.sh file now enables you to remove the ep-config.xml file, a specific version of the SAS Embedded Process, or all versions of the SAS Embedded Process.
- A new panel that lists the products being installed by the SAS Deployment Manager has been added. Information that appears on other panels also reflects more closely what is being installed.
July 2015 Release of SAS 9.4: Changes and Enhancements

The installation and configuration of the SAS Embedded Process for Hadoop has changed.

- For Cloudera and Hortonworks, Cloudera Manager and Ambari are used to install the SAS Embedded Process and the SAS Hadoop MapReduce JAR files.
- For IBM BigInsights, MapR, and Pivotal HD, the in-database deployment package is delivered to the client from the SAS Install Depot.
- The SAS Embedded Process and the SAS Hadoop MapReduce JAR files are installed with one script instead of in two separate scripts. The new process has a smaller client footprint and is a faster install process.
- The sasep-servers.sh file has changed names to the sasep-admin.sh file. Some of the scripts arguments are no longer needed and have been deleted. Other arguments have been added.

August 2014 Release of SAS 9.4: Changes and Enhancements

In the August 2014 release of SAS 9.4, the following changes and enhancements were made:

- Instead of manually selecting the Hadoop JAR files to the client machine, the SAS Embedded Process determines which version of the JAR files are required and gathers them into a ZIP file for you to copy to the client machine.
- You now have the option whether to automatically start the SAS Embedded Process when the installation is complete.

April 2014 Release of SAS 9.4: Changes and Enhancements

The documentation about the installation and configuration of the SAS Embedded Process was enhanced.

December 2013 Release of SAS 9.4: Changes and Enhancements

In the December 2013 release of SAS 9.4, the following changes and enhancements were made:

- The trace log messages for the SAS Embedded Process are now stored in the MapReduce job log.
- A new option, hdfsuser, is available in the sasep-servers.sh script. hdfsuser specifies the user ID that has Write access to HDFS root directory.
- The Cloudera JAR files for the SAS Embedded Process have been replaced by a set of Apache JAR files. The new JAR files are based on a release of the Apache Hadoop instead of a particular Hadoop distributor.

SAS 9.4: Changes and Enhancements

The Hadoop scripts that install, control, and provide a status of the SAS Embedded Process have changed. There is now just one script, sasep-servers.sh, that installs both
the SAS Embedded Process and the Hadoop JAR files. Running this script also enables you to start, stop, and provide a status of the SAS Embedded Process.

---

**Netezza Changes**

**November 2016 Release of SAS 9.4: Changes and Enhancements**

The installation and configuration of the SAS Embedded Process for Netezza has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.

**July 2015 Release of SAS 9.4: Changes and Enhancements**

The SAS Embedded Process for Netezza has a new cartridge file that creates the NZRC database.

**SAS 9.4: Changes and Enhancements**

In-database scoring for Netezza has been enhanced by the addition of the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Netezza to read and write data. The SAS Embedded Process can be used with the SAS Scoring Accelerator for Netezza to run scoring models.

---

**Oracle Changes**

**November 2016 Release of SAS 9.4: Changes and Enhancements**

The installation and configuration of the SAS Embedded Process for Oracle has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.

---

**SAP HANA Changes**

**November 2016 Release of SAS 9.4: Changes and Enhancements**

The installation and configuration of the SAS Embedded Process for SAP has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.
July 2015 Release of SAS 9.4: Changes and Enhancements

If you are using SAP HANA SPS9, the SAS Embedded Process for SAP HANA must be manually started. For previous versions, the SAS Embedded Process was automatically started by the SASAFL plug-in. In addition, a different procedure must be used to deploy the SASAFL plug-in.

SPD Server Changes

November 2018 Release of SAS 9.4: Changes and Enhancements

In-database scoring for the SAS Scalable Performance Data Server is no longer supported.

Teradata Changes

November 2018 Release of SAS 9.4: Changes and Enhancements

Due to changes to the deployment of the SAS Embedded Process, install scripts for SAS Data Quality Accelerator for Teradata stored processes are delivered in a different location.

November 2016 Release of SAS 9.4: Changes and Enhancements

SAS in-database processing in Teradata now supports single sign-on (SSO) authentication.

July 2015 Release of SAS 9.4: Changes and Enhancements

The installation and configuration of the SAS Embedded Process for Teradata has changed. The in-database deployment package is delivered to the client from the SAS Install Depot. The new process has a smaller client footprint and is a faster install process.
SAS Deployment Manager

**November 2018 Release of SAS 9.4: Changes and Enhancements**

In the November 2018 release of SAS 9.4, the SAS Deployment Manager is no longer used to deploy the SAS Embedded Process for Hadoop. The SAS Embedded Process deployment package is now being delivered from a SAS Viya repository and the deployment process has changed. It is the same for both SAS 9.4 and SAS Viya. The documentation is now available in *SAS Embedded Process: Deployment Guide*.

The SAS Deployment Manager can continue to be used to deploy the SAS Data Quality Accelerator, the SAS Data Loader for Hadoop Spark Engine, and the SAS Quality Knowledge Base, if you license the SAS Data Loader for Hadoop. For more information, see *SAS Data Loader for Hadoop: Installation and Configuration*.

**September 2017 Release of SAS 9.4: Changes and Enhancements**

In the September 2017 release of SAS 9.4, the following changes and enhancements were made:

- The SAS Deployment Manager can now be run if the cluster manager is enabled with Transport Layer Security (TLS). The SAS Deployment Manager validates if the Certificate of Authority (CA) exists for the host that you are trying to access. If the CA does not exist, a warning message is issued and asks to run another SAS Deployment Manager task to add the CA. There is also a SAS Deployment Manager task to remove the CA.

  *Note:* All discussion of TLS is also applicable to the predecessor protocol, Secure Sockets Layer (SSL).

- If you license SAS Data Loader for Hadoop, you can now deploy the SAS Embedded Process and the SAS Data Quality Accelerator, the SAS Data Loader for Hadoop Spark Engine, and the SAS Quality Knowledge Base individually, all together, or in any combination at one time.
Overview of SAS In-Database Products

SAS in-database products integrate SAS solutions, SAS analytic processes, and third-party database management systems. Using SAS in-database technology, you can run scoring models, some SAS procedures, DS2 threaded programs, and formatted SQL queries inside the database. When using conventional processing, all rows of data are returned from the database to SAS. When using SAS in-database technology, processing is done inside the database and thus does not require the transfer of data.

To perform in-database processing, the following SAS products require additional installation and configuration:

- SAS/ACCESS Interface to Aster, SAS/ACCESS Interface to DB2, SAS/ACCESS Interface to Greenplum, SAS/ACCESS Interface to Hadoop, SAS/ACCESS Interface to Netezza, SAS/ACCESS Interface to Oracle, SAS/ACCESS Interface to SAP HANA, and SAS/ACCESS Interface to Teradata
  
  The SAS/ACCESS interfaces to the individual databases include components that are required for both format publishing to the database and for running some Base SAS procedures inside the database.

- SAS Scoring Accelerator for Aster, SAS Scoring Accelerator for DB2, SAS Scoring Accelerator for Greenplum, SAS Scoring Accelerator for Hadoop, SAS Scoring Accelerator for Netezza, SAS Scoring Accelerator for Oracle, SAS Scoring Accelerator for SAP HANA, and SAS Scoring Accelerator for Teradata

- SAS In-Database Code Accelerator for Greenplum, SAS In-Database Code Accelerator for Hadoop, and SAS In-Database Code Accelerator for Teradata

- SAS Contextual Analysis In-Database Scoring for Hadoop
• SAS Analytics Accelerator for Teradata
• SAS Data Loader for Hadoop
• SAS Data Quality Accelerator for Teradata
• SAS Model Manager In-Database Scoring Scripts

*Note:* In SAS Viya 3.3 and later, you can publish and score both DS2 and DATA step models on the CAS server. In addition, you can copy or publish the DS2 and DATA step models to either Hadoop or Teradata. You can then score the model programs on the data source using the SAS Embedded Process and the SAS Data Connect Accelerator. For more information about scoring in SAS Viya, see *SAS DS2 Programmer’s Guide.*

---

**What Is Covered in This Document?**

**Overview**

The instructions that you use to deploy and configure components that are needed for in-database processing depends on the software that you have installed.

This document contains instructions for:

- deploying the in-database deployment package for DB2, Greenplum, Netezza, and Teradata
- running macros that are needed to register other SAS in-database components
- configuring the SAS Embedded Process

*Note:* Starting with SAS 9.4M6, the information for deploying the SAS Embedded Process for all data sources has been removed from this document. The SAS Embedded Process deployment package is now being delivered from one location and the deployment is the same for both SAS 9.4 and SAS Viya. The documentation is now available in *SAS Embedded Process: Deployment Guide.*

This document is intended for the system administrator, the database administrator, or both. It is expected that you work closely with the SAS programmers who use these products.

This document is divided by database management systems.

**Instructions for the SAS Scoring Accelerator, the SAS Analytics Accelerator, the SAS Data Quality Accelerator for Teradata, and the SAS In-Database Code Accelerator**

This document, *SAS In-Database Products: Administrator’s Guide,* contains the instructions for deploying the in-database deployment package when using SAS Scoring Accelerator, SAS Analytics Accelerator, SAS Data Quality Accelerator for Teradata, and SAS In-Database Code Accelerator. These components are contained in a deployment package that is specific to your database.

The name and version of the in-database deployment packages are as follows:

- SAS Formats Library for DB2 3.1
- SAS Formats Library for Greenplum 3.1
• SAS Formats Library for Netezza 3.1
• SAS Formats Library for Teradata 3.1

Note: If you use SAS Model Manager for in-database scoring with DB2, Greenplum, Hadoop, Netezza, or Teradata, additional configuration tasks are needed. This document provides detailed instructions for configuring a database for use with SAS Model Manager.

Instructions for the SAS Embedded Process

Starting with SAS 9.4M6, the information for deploying the SAS Embedded Process for all data sources has been removed from this document. The SAS Embedded Process deployment package is now being delivered from one location and the deployment is the same for both SAS 9.4 and SAS Viya. The documentation is now available in SAS Embedded Process: Deployment Guide.

Instructions for SAS Viya

If your data source is Teradata or Hadoop and you are using SAS Viya, refer to SAS Embedded Process: Deployment Guide and SAS Viya for Linux: Deployment Guide for additional configuration instructions for the SAS Embedded Process and Data Connect Accelerator.

Instructions for SAS Data Loader for Hadoop

If your software product is SAS Data Loader for Hadoop, refer to SAS Embedded Process: Deployment Guide and SAS Data Loader for Hadoop: Installation and Configuration Guide for instructions on deploying the SAS Embedded Process and additional configuration instructions, respectively.

Instructions for SAS Contextual Analytics In-Database Scoring for Hadoop

If your software product is SAS Contextual Analytics In-Database Scoring for Hadoop, refer to SAS Embedded Process: Deployment Guide and SAS Contextual Analytics In-Database Scoring for Hadoop: Administrator’s Guide for instructions on deploying the SAS Embedded Process and additional configuration instructions, respectively.
Part 2

Administrator’s Guide for Aster

Chapter 2
Aster Administrator's Guide
Prerequisites

In-database processing for Aster requires the following:

• SAS Foundation and the SAS/ACCESS Interface to Aster. This software must be installed before you configure in-database processing for ASTER.

• a specific version of the Aster client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

• the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Aster to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your Aster system so that the SAS_SCORE( ) and the SAS_PUT( ) functions can access the routines within its run-time libraries. For more information, see SAS Embedded Process: Deployment Guide.

Overview of the SAS Embedded Process for Aster

The SAS Embedded Process for Aster must be installed and configured before you can use the %INDAC_PUBLISH_MODEL scoring publishing macro to create scoring files inside the database and the %INDAC_PUBLISH_FORMATS format publishing macro to create user-defined format files.

For more information about using the scoring and format publishing macros, see the SAS In-Database Products: User’s Guide.
Aster Permissions

The person who installs the SAS Embedded Process for Aster needs root privileges for the queen node. This permission is most likely, but not necessarily, needed by the Aster system administrator.

For Aster 6.10 or later, the following schema permissions are needed by the person who runs the scoring and format publishing macros, because all functions and files can be published to a specific schema.

**USAGE permission**

```sql
GRANT USAGE ON SCHEMA your-schema-name TO your-user-id;
```

**INSTALL FILE permission**

```sql
GRANT INSTALL FILE ON SCHEMA your-schema-name TO your-user-id;
```

**CREATE permission**

```sql
GRANT CREATE ON SCHEMA your-schema-name TO your-user-id;
```

**EXECUTE permission**

```sql
GRANT EXECUTE ON FUNCTION PUBLIC.SAS_SCORE TO your-user-id;
GRANT EXECUTE ON FUNCTION PUBLIC.SAS_PUT TO your-user-id;
GRANT EXECUTE ON FUNCTION PUBLIC.SAS_ROW TO your-user-id;
GRANT EXECUTE ON FUNCTION PUBLIC.SAS_PARTITION TO your-user-id;
```

Documentation for Using In-Database Processing in Aster

For information about how to publish SAS formats and scoring models, see the *SAS In-Database Products: User’s Guide*. 
Part 3

Administrator’s Guide for DB2

Chapter 3

DB2 Administrator’s Guide
Chapter 3
DB2 Administrator’s Guide

Prerequisites

The in-database deployment package for DB2 requires the following:

• SAS Foundation and the SAS/ACCESS Interface to DB2. This software must be installed before you install and configure the in-database deployment package for DB2.

• a specific version of the DB2 client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

• Permissions to publish the SAS_COMPILEUDF and SAS_DELETEUDF functions and create the SASUDF_COMPILER_PATH and SASUDF_DB2PATH global variables.

For more information, see “DB2 Permissions” on page 28.

• the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within DB2 to read and write data. The SAS Embedded Process contains
Overview of the In-Database Deployment Package for DB2

The in-database deployment package for DB2 contains the SAS formats library and the precompiled binary files for two additional utility functions. The in-database deployment package is available for installation after you deploy SAS/ACCESS Interface to DB2.

The in-database deployment package for DB2 must be installed and configured before you can perform the following tasks:

- Use the `%INDB2_PUBLISH_FORMATS` format publishing macro to create or publish the SAS_PUT() function and to create or publish user-defined formats as format functions inside the database.
- Use the `%INDB2_PUBLISH_MODEL` scoring publishing macro to create scoring model functions inside the database.

For more information about using the format and scoring publishing macros, see the *SAS In-Database Products: User’s Guide*.

The SAS formats library is a run-time library that is installed on your DB2 system so that the SAS scoring model functions and the SAS_PUT() function created in DB2 can access the routines within the run-time library. The SAS formats library contains the formats that are supplied by SAS.

The two publishing macros, `%INDB2_PUBLISH_COMPILEUDF` and `%INDB2_PUBLISH_DELETEUDF`, register the two utility functions in the database. The utility functions are called by the format and scoring publishing macros. You must run these two macros before you run the format and scoring publishing macros.

Function Publishing Process in DB2

To publish scoring model functions and the SAS_PUT() function on a DB2 server, the publishing macros perform the following tasks:

- Create and transfer the files to the DB2 environment.
- Compile those source files into object files using the appropriate compiler for that system.
- Link with the SAS formats library.

After that, the publishing macros register the format and scoring model functions in DB2 with those object files. If an existing format or scoring model function is replaced, the publishing macros remove the obsolete object file upon successful compilation and publication of the new format or scoring model functions.

The publishing macros use a SAS FILENAME SFTP statement to transfer the format or scoring source files to the DB2 server. An SFTP statement offers a secure method of user validation and data transfer. The SAS FILENAME SFTP statement dynamically launches an SFTP or PSFTP executable, which creates an SSH client process that creates macros, run-time libraries, and other software that is installed on your DB2 system. For more information, see *SAS Embedded Process: Deployment Guide*. 
a secure connection to an OpenSSH Server. All conversation across this connection is encrypted, from user authentication to the data transfers.

Currently, only the OpenSSH client and server on UNIX that supports protocol level SSH-2 and the PUTTY client on WINDOWS, are supported. For more information about setting up the SSH software to enable the SAS SFTP to work, please see Setting Up SSH Client Software in UNIX and Windows Environments for Use with the SFTP Access Method in SAS 9.2, SAS 9.3, and SAS 9.4, located at http://support.sas.com/techsup/technote/ts800.pdf.

Note: This process is valid only when using publishing formats and scoring functions. It is not applicable to the SAS Embedded Process. If you use the SAS Embedded Process, the scoring publishing macro creates the scoring files and uses the SAS/ACCESS Interface to DB2 to insert the scoring files into a model table.

DB2 Installation and Configuration

DB2 Installation and Configuration Steps

This section describes how to install and configure the in-database deployment package for DB2 (SAS Formats Library for DB2 and binary files). The in-database deployment package is available for installation after you deploy SAS/ACCESS Interface to DB2.

1. If you are upgrading from a previous version, follow the instructions in “Upgrading from a Previous Version” on page 16 before installing the in-database deployment package.

2. Verify that you can use PSFTP from Windows to UNIX without being prompted for a password or cache.

   To do this, enter the following commands from the PSFTP prompt, where userid is the user ID that you want to log on as and machinename is the machine to which you want to log on.

   psftp> open userid@machinename
   psftp> ls

3. Install the SAS formats library, the binary files for the SAS_COMPILEUDF, and SAS_DELETEUDF functions.

   For more information, see “Installing the SAS Formats Library and Binary Files to DB2” on page 16.

4. Run the %INDB2_PUBLISH_COMPILEUDF macro to create the SAS_COMPILEUDF function.

   For more information, see “Running the %INDB2_PUBLISH_COMPILEUDF Macro” on page 18.

5. Run the %INDB2_PUBLISH_DELETEUDF macro to create the SAS_DELETEUDF function.

   For more information, see “Running the %INDB2_PUBLISH_DELETEUDF Macro” on page 22.

6. If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, perform the additional configuration tasks provided in Chapter 12, “Configuring SAS Model Manager,” on page 115.
Upgrading from a Previous Version

To upgrade a previous version of the SAS formats library and binary files, follow these steps:

1. Drop the SAS_COMPILEUDF and SAS_DELETEUDF functions by running the %INDB2_PUBLISH_COMPILEUDF and %INDB2_PUBLISH_DELETEUDF macros with ACTION=DROP.

Here is an example:

```
%let indconn = user=abcd password=xxxx database=indbdb server=indbsvr;
%indb2_publish_compileudf(action=drop, db2path=/users/db2v10/sqlib, compiler_path=/usr/vac/bin);
%indb2_publish_deleteudf(action=drop);
```

2. Confirm that the SAS_COMPILEUDF and SAS_DELETEUDF functions were dropped.

Here is an example:

```
proc sql noerrorstop;
   connect to db2 (user=abcd password=xxxx database=indbdb);
   select * from connection to db2 (select cast(funcname as char(40)),
   cast(definer as char(20)) from syscat.functions
   where funcschema='SASLIB' );
quit;
```

3. Stop the DB2 instance.

   a. Log on to the DB2 server and enter this command to determine whether there are any users connected to the instance:

      `db2 list applications`

   b. If any users are connected, enter these commands to force them off before the instance is stopped and clear any background processes:

      `db2 force applications all`
      `db2 terminate`

   c. Enter this command to stop the DB2 instance:

      `db2stop`

4. Enter the following commands to move to the `db2instancepath/sqlib/function` directory and remove the SAS directory. `db2instancepath/sqlib/function` is the path to the SAS_COMPILEUDF and SAS_DELETEUDF functions in the DB2 instance.

```
    cd db2instancepath/sqlib/function
    rm -fr SAS
```

Installing the SAS Formats Library and Binary Files to DB2

Move the SAS Formats and Binary Files to DB2

The SAS formats library and the binary files that need to be moved to DB2 are contained in a self-extracting archive file. You can use PSFTP, SFTP, or FTP to transfer the file to the DB2 server to be unpacked and compiled.
The self-extracting archive file contains the SAS formats library and the binary files for the SAS_COMPILEUDF and SAS_DELETEUDF functions. You need these files when you want to use scoring functions to run your scoring model and when publishing SAS formats.

This self-extracting archive file is located in the `YourSASHome/SASFormatsLibraryforDB2/3.1/DB2on<AIX|Linux64>/` directory.

Choose the self-extracting archive file based on the UNIX platform that your DB2 server runs on.

- AIX: `acceldb2fmt-3.1-n_r64.sh`
- Linux(x86_64): `acceldb2fmt-3.1-n_lax.sh`

`n` is a number that indicates the latest version of the file. If this is the initial installation, `n` has a value of 1. Each time you re-install or upgrade, `n` is incremented by 1.

The file does not have to be downloaded to a specific location. However, you need to note where it is downloaded so that it can be executed as the DB2 instance owner at a later time. It is recommended that you put the `acceldb2fmt` file somewhere other than the DB2 home directory tree.

List the directory in UNIX to verify that the files have been moved.

**Unpack and Install the SAS Formats Library and Binary Files for DB2**

After the `acceldb2fmt-3.1-n_lax.sh` or `acceldb2fmt-3.1-n_r64.sh` self-extracting archive file is transferred to the DB2 machine, follow these steps to unpack the file. `n` is a number that indicates the latest version of the file. If this is the initial installation, `n` has a value of 1. Each time you re-install or upgrade, `n` is incremented by 1.

1. Log on as the user who owns the DB2 instance from a secured shell, such as SSH.
2. Change to the directory where you put the `acceldb2fmt` file.
   ```
   cd path_to_sh_file
   path_to_sh_file is the location to which you copied the self-extracting archive file.
   ```
3. Change permissions on the file to enable you to execute the script.
   ```
   chmod 755 -r acceldb2fmt-3.1-n_r64.sh
   ```
   **Note:** AIX is the platform that is being used as an example for all the steps in this topic.
4. If there is a previously created self-extracting archive file in the SAS directory, you must either rename or remove the directory. These are examples of the commands that you would use:
   ```
   mv SAS to SAS_OLD /* renames the SAS directory */
   rm -fr SAS /* removes the SAS directory */
   ```
5. Use the following commands to unpack the appropriate self-extracting archive file.
   ```
   ./sh_file
   sh_file is either acceldb2fmt-3.1-n_lax.sh or acceldb2fmt-3.1-n_r64.sh depending on your platform.
   ```
   After this script is run and the files are unpacked, a SAS tree is built in the current directory. The content of the target directories should be similar to the following,
depending on your operating system. Part of the directory path is shaded to emphasize the different target directories that are used.

/path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-n/bin/InstallAccelDB2Fmt.sh
/path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-n/bin/CopySASFiles.sh
/path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-n/lib/SAS_CompileUDF
/path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-n/lib/SAS_DeleteUDF
/path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-n/lib/libjazxfbrs.so
/path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-3.1-n

6. Use the following command to place the files in the DB2 instance:

```
path_to_sh_file/SAS/SASFormatsLibraryForDB2/3.1-n/bin/
  CopySASFiles.sh db2instancepath/sqllib
```

`db2instancepath/sqllib` is the path to the `sqllib` directory of the DB2 instance that you want to use.

After this script is run and the files are copied, the target directory should look similar to this.

```
db2instancepath/sqllib/function/SAS/SAS_CompileUDF
db2instancepath/sqllib/function/SAS/SAS_DeleteUDF
db2instancepath/sqllib/function/SAS/libjazxfbrs.so
```

**Note:** If the SAS_CompileUDF, SAS_DeleteUDF, and libjazxfbrs.so files currently exist under the target directory, you must rename the existing files before you run the CopySASFiles.sh script. Otherwise, the CopySASFiles.sh script does not work, and you get a "Text file is busy" message for each of the three files.

7. Use the DB2SET command to tell DB2 where to find the 64-bit formats library.

```
db2set DB2LIBPATH="db2instancepath/sqllib/function/SAS"
```

`db2instancepath/sqllib` is the path to the `sqllib` directory of the DB2 instance that you want to use.

The DB2 instance owner must run this command for it to be successful. Note that this is similar to setting a UNIX system environment variable using the UNIX EXPORT or SETENV commands. DB2SET registers the environment variable within DB2 only for the specified database server.

For more information on DB2SET, see “DB2SET Command Syntax for In-Database Processing” on page 26.

8. To verify that DB2LIBPATH was set appropriately, run the DB2SET command without any parameters.

```
db2set
```

The results should be similar to this one if it was set correctly.

```
DB2LIBPATH=db2instancepath/sqllib/function/SAS
```

---

**Running the %INDB2_PUBLISH_COMPILEUDF Macro**

**Overview of the %INDB2_PUBLISH_COMPILEUDF Macro**

The %INDB2_PUBLISH_COMPILEUDF macro publishes the following components to the SASLIB schema in a DB2 database:

- SAS_COMPILEUDF function
The SAS_COMPILEUDF function facilitates the %INDB2_PUBLISH_FORMATS format publishing macro and the %INDB2_PUBLISH_MODEL scoring publishing macro when you use scoring functions to run the scoring model. The SAS_COMPILEUDF function performs the following tasks:

• compiles the format and scoring model source files into object files. This compilation occurs through the SQL interface using an appropriate compiler for the system.

• links with the SAS formats library that is needed for format and scoring model publishing.

• copies the object files to the db2instancepath/sqlib/function/SAS directory. You specify the value of db2instancepath in the %INDB2_PUBLISH_COMPILEUDF macro syntax.

• SASUDF_DB2PATH and SASUDF_COMPILER_PATH global variables

The SASUDF_DB2PATH and the SASUDF_COMPILER_PATH global variables are used when you publish the format and scoring model functions.

You have to run the %INDB2_PUBLISH_COMPILEUDF macro only one time in a given database.

The SAS_COMPILEUDF function must be published before you run the %INDB2_PUBLISH_DELETEUDF macro, the %INDB2_PUBLISH_FORMATS macro, and the %INDB2_PUBLISH_MODEL macro. Otherwise, these macros fail.

Note: To publish the SAS_COMPILEUDF function, you must have the appropriate DB2 user permissions to create and execute this function in the SASLIB schema and in the specified database. For more information, see “DB2 Permissions” on page 28.

%INDB2_PUBLISH_COMPILEUDF Macro Run Process

To run the %INDB2_PUBLISH_COMPILEUDF macro, follow these steps:

1. Create a SASLIB schema in the database where the SAS_COMPILEUDF function is to be published.

   The SASLIB schema is used when publishing the %INDB2_PUBLISH_COMPILEUDF macro for DB2 in-database processing.

   You specify that database in the DATABASE argument of the %INDB2_PUBLISH_COMPILEUDF macro. For more information, see “%INDB2_PUBLISH_COMPILEUDF Macro Syntax” on page 21.

   The SASLIB schema contains the SAS_COMPILEUDF and SAS_DELETEUDF functions and the SASUDF_DB2PATH and SASUDF_COMPILER_PATH global variables.

2. Start SAS and submit the following command in the Enhanced Editor or Program Editor:

   ```sas
   %let indconn = server=yourserver user=youruserid password=yourpwd
database=yourdb schema=saslib;
   ```

   For more information, see the “INDCONN Macro Variable” on page 20.

3. Run the %INDB2_PUBLISH_COMPILEUDF macro. For more information, see “%INDB2_PUBLISH_COMPILEUDF Macro Syntax” on page 21.

You can verify that the SAS_COMPILEUDF function and global variables have been published successfully. For more information, see “Validating the Publishing of
SAS_COMPILEUDF and SAS_DELETEUDF Functions and Global Variables” on page 25.

After the SAS_COMPILEUDF function is published, run the %INDB2_PUBLISH_DELETEUDF publishing macro to create the SAS_DELETEUDF function. For more information, see “Running the %INDB2_PUBLISH_DELETEUDF Macro” on page 22.

**INDCONN Macro Variable**

The INDCONN macro variable provides the credentials to make a connection to DB2. You must specify the server, user, password, and database information to access the machine on which you have installed the DB2 database. You must assign the INDCONN macro variable before the %INDB2_PUBLISH_COMPILEUDF macro is invoked.

The value of the INDCONN macro variable for the %INDB2_PUBLISH_COMPILEUDF macro has this format.

```
SERVER=server USER=userid PASSWORD=password DATABASE=database <SCHEMA=SASLIB>
```

**SERVER=server | 'server'**

specifies the DB2 server name or the IP address of the server host. If the server name contains spaces or nonalphanumeric characters, enclose the server name in quotation marks.

**Requirement**

The name must be consistent with how the host name was cached when PSFTP server was run from the command window. If the full server name was cached, you must use the full server name in the SERVER argument. If the short server name was cached, you must use the short server name. For example, if the long name, `disk3295.unx.comp.com`, is used when PSFTP was run, then `server=disk3295.unx.comp.com` must be specified. If the short name, `disk3295`, was used, then `server=disk3295` must be specified. For more information, see “DB2 Installation and Configuration Steps” on page 15.

**USER=userid | 'userid'**

specifies the DB2 user name (also called the user ID) that is used to connect to the database. If the user name contains spaces or nonalphanumeric characters, enclose the user name in quotation marks.

**PASSWORD=password | 'password'**

specifies the password that is associated with your DB2 user ID. If the password contains spaces or nonalphanumeric characters, enclose the password in quotation marks.

**Tip** Use only PASSWORD=, PASS=, or PW= for the password argument. PWD= is not supported and causes an error.

**DATABASE=database | 'database'**

specifies the DB2 database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, enclose the database name in quotation marks.

**Requirement**

The SAS_COMPILEUDF function is created as a Unicode function. If the database is not a Unicode database, then the alternate collating sequence must be configured to use `identity_16bit`. 
**SCHEMA=SASLIB**

specifies SASLIB as the schema name.

**Default**  
SASLIB

**Restriction**  
The SAS_COMPILEUDF function and the two global variables (SASUDF_DB2PATH and SASUDF_COMPILER_PATH) are published to the SASLIB schema in the specified database. If a value other than SASLIB is used, it is ignored.

**Requirement**  
The SASLIB schema must be created before publishing the SAS_COMPILEUDF and SASDELETEUDF functions.

---

**%INDB2_PUBLISH_COMPILEUDF Macro Syntax**

```plaintext
%INDB2_PUBLISH_COMPILEUDF
    (DB2PATH=db2instancepath/sqllib
     , COMPILER_PATH=compiler-path-directory
     <, DATABASE=database-name>
     <, ACTION=CREATE | REPLACE | DROP>
     <, OBJNAME=object-file-name>
     <, OUTDIR=diagnostic-output-directory>
    );
```

**Arguments**

- **DB2PATH=db2instancepath/sqllib**  
specifies the parent directory that contains the function/SAS subdirectory, where all the object files are stored and defines the SASUDF_DB2PATH global variable that is used when publishing the format and scoring model functions.

  **Interaction**  
  `db2instancepath` should be the same path as the path that was specified during the installation of the SAS_COMPILEUDF binary file. For more information, see Step 3 in “Unpack and Install the SAS Formats Library and Binary Files for DB2” on page 17.

  **Tip**  
The SASUDF_DB2PATH global variable is defined in the SASLIB schema under the specified database name.

- **COMPILER_PATH=compiler-path-directory**  
specifies the path to the location of the compiler that compiles the source files and defines the SASUDF_COMPILER_PATH global variable that is used when publishing the format and scoring model functions.

  **Tip**  
The SASUDF_COMPILER_PATH global variable is defined in the SASLIB schema under the specified database name. The XLC compiler should be used for AIX, and the GGG compiler should be used for Linux.

- **DATABASE=database-name**  
specifies the name of a DB2 database to which the SAS_COMPILEUDF function is published.

  **Interaction**  
The database that you specify in the DATABASE= argument takes precedence over the database that you specify in the INDCONN macro variable. For more information, see “%INDB2_PUBLISH_COMPILEUDF Macro Run Process” on page 19.

- **ACTION=CREATE | REPLACE | DROP**  
specifies that the macro performs one of the following actions:
CREATE
   creates a new SAS_COMPILEUDF function.

REPLACE
   overwrites the current SAS_COMPILEUDF function, if a SAS_COMPILEUDF
   function by the same name is already registered, or creates a new
   SAS_COMPILEUDF function if one is not registered.

DROP
   causes the SAS_COMPILEUDF function to be dropped from the DB2 database.

Default  CREATE
Tip     If the SAS_COMPILEUDF function was published previously and you
         now specify ACTION=CREATE, you receive warning messages from
         DB2. If the SAS_COMPILEUDF function was published previously and
         you specify ACTION=REPLACE, no warnings are issued.

OBJNAME=object-file-name
   specifies the object file name that the publishing macro uses to register the
   SAS_COMPILEUDF function. The object file name is a file system reference to a
   specific object file, and the value entered for OBJNAME must match the name as it
   exists in the file system. For example, SAS_CompileUDF is mixed case.

Default  SAS_CompileUDF
Interaction If the SAS_COMPILEUDF function is updated, you might want to
             rename the object file to avoid stopping and restarting the database. If
             so, the SAS_COMPILEUDF function needs to be reregistered with the
             new object file name.

OUTDIR=diagnostic-output-directory
   specifies a directory that contains diagnostic files.

Tip     Files that are produced include an event log that contains detailed information
         about the success or failure of the publishing process.

Running the %INDB2_PUBLISH_DELETEUDF Macro

Overview of the %INDB2_PUBLISH_DELETEUDF Macro
The %INDB2_PUBLISH_DELETEUDF macro publishes the SAS_DELETEUDF
function in the SASLIB schema of a DB2 database. The SAS_DELETEUDF function
facilitates the %INDB2_PUBLISH_FORMATS format publishing macro and the
%INDB2_PUBLISH_MODEL scoring publishing macro. The SAS_DELETEUDF function removes existing object files when the format or scoring publishing macro
registers new ones by the same name.

You have to run the %INDB2_PUBLISH_DELETEUDF macro only one time in a given
database.

The SAS_COMPILEUDF function must be published before you run the
%INDB2_PUBLISH_DELETEUDF macro, the %INDB2_PUBLISH_FORMATS
macro, and the %INDB2_PUBLISH_MODEL macro. Otherwise, these macros fail.

Note: To publish the SAS_DELETEUDF function, you must have the appropriate DB2
user permissions to create and execute this function in the SASLIB schema and
specified database. For more information, see “DB2 Permissions” on page 28.
To run the %INDB2_PUBLISH_DELETEUDF macro, follow these steps:

1. Ensure that you have created a SASLIB schema in the database where the SAS_DELETEUDF function is to be published.

   Use the SASLIB schema when publishing the %INDB2_PUBLISH_DELETEUDF macro for DB2 in-database processing.

   The SASLIB schema should have been created before you ran the %INDB2_PUBLISH_COMPILEUDF macro to create the SAS_COMPILEUDF function. The SASLIB schema contains the SAS_COMPILEUDF and SAS_DELETEUDF functions and the SASUDF_DB2PATH and SASUDF_COMPILER_PATH global variables.

   The SAS_COMPILEUDF function must be published before you run the %INDB2_PUBLISH_DELETEUDF macro. The SAS_COMPILEUDF and SAS_DELETEUDF functions must be published to the SASLIB schema in the same database. For more information about creating the SASLIB schema, see “%INDB2_PUBLISH_COMPILEUDF Macro Run Process” on page 19.

2. Start SAS and submit the following command in the Enhanced Editor or Program Editor.

   ```sas
   %let indconn = server=yourserver user=youruserid password=yourpwd
database=yourdb schema=saslib;
   ```

   For more information, see the “INDCONN Macro Variable” on page 23.

3. Run the %INDB2_PUBLISH_DELETEUDF macro. For more information, see “%INDB2_PUBLISH_DELETEUDF Macro Syntax” on page 24.

   You can verify that the function has been published successfully. For more information, see “Validating the Publishing of SAS_COMPILEUDF and SAS_DELETEUDF Functions and Global Variables” on page 25.

   After the SAS_DELETEUDF function is published, the %INDB2_PUBLISH_FORMATS and the %INDB2_PUBLISH_MODEL macros can be run to publish the format and scoring model functions.

### INDCONN Macro Variable

The INDCONN macro variable provides the credentials to make a connection to DB2. You must specify the server, user, password, and database information to access the machine on which you have installed the DB2 database. You must assign the INDCONN macro variable before the %INDB2_PUBLISH_DELETEUDF macro is invoked.

The value of the INDCONN macro variable for the %INDB2_PUBLISH_DELETEUDF macro has this format.

```
SERVER=server USER=userid PASSWORD=password
DATABASE=database <SCHEMA=SASLIB>
```

**SERVER=server**

specifies the DB2 server name or the IP address of the server host. If the server name contains spaces or nonalphanumeric characters, enclose the server name in quotation marks.

**Requirement**

The name must be consistent with how the host name was cached when PSFTP `server` was run from the command window. If the full server name was cached, use the full server name in the SERVER argument. If the short server name was cached, use the short server name. For example, if the long name, `disk3295.unx.comp.com`, is
used when PSFTP was run, then server=disk3295.unx.comp.com
must be specified. If the short name, disk3295, was used, then
server=disk3295 must be specified. For more information, see “DB2
Installation and Configuration Steps” on page 15.

**USER=userid**
specifies the DB2 user name (also called the user ID) that is used to connect to the
database. If the user name contains spaces or nonalphanumeric characters, enclose
the user name in quotation marks.

**PASSWORD=password**
specifies the password that is associated with your DB2 user ID. If the password
contains spaces or nonalphabetic characters, enclose the password in quotation
marks.

**Tip** Use only PASSWORD=, PASS=, or PW= for the password argument. PWD= is
not supported and causes errors.

**DATABASE=database**
specifies the DB2 database that contains the tables and views that you want to
access. If the database name contains spaces or nonalphanumeric characters, enclose
the database name in quotation marks.

**SCHEMA=SASLIB**
specifies SASLIB as the schema name.

**Default** SASLIB

**Restriction** The SAS_DELETEUDF function is published to the SASLIB schema
in the specified database. If a value other than SASLIB is used, it is
ignored.

**Requirement** Create the SASLIB schema before publishing the
SAS_COMPILEUDF and SAS_DELETEUDF functions.

---

**%INDB2_PUBLISH_DELETEUDF Macro Syntax**

```
%INDB2_PUBLISH_DELETEUDF
(<DATABASE=database-name>
 , ACTION=CREATE | REPLACE | DROP>
 , OUTDIR=diagnostic-output-directory>
);
```

**Arguments**

**DATABASE=database-name**
specifies the name of a DB2 database to which the SAS_DELETEUDF function is
published.

**Interaction** The database that you specify in the DATABASE argument takes
precedence over the database that you specify in the INDCONN macro
variable. For more information, see “Running the
%INDB2_PUBLISH_DELETEUDF Macro” on page 22.

**ACTION=CREATE | REPLACE | DROP**
specifies that the macro performs one of the following actions:

**CREATE**
creates a new SAS_DELETEUDF function.
REPLACE
  overwrites the current SAS_DELETEUDF function, if a SAS_DELETEUDF function by the same name is already registered, or creates a new SAS_DELETEUDF function if one is not registered.

DROP
  causes the SAS_DELETEUDF function to be dropped from the DB2 database.

Default  CREATE

Tip  If the SAS_DELETEUDF function was published previously and you specify ACTION=CREATE, you receive warning messages from DB2. If the SAS_DELETEUDF function was published previously and you specify ACTION=REPLACE, no warnings are issued.

OUTDIR=diagnostic-output-directory
  specifies a directory that contains diagnostic files.

Tip  Files that are produced include an event log that contains detailed information about the success or failure of the publishing process.

Validating the Publishing of SAS_COMPILEUDF and SAS_DELETEUDF Functions and Global Variables

To validate that the SAS_COMPILEUDF and SAS_DELETEUDF functions and global variables are created properly, follow these steps.

1. Connect to your DB2 database using Command Line Processor (CLP).

2. Enter the following command to verify that the SASUDF_COMPILER_PATH global variable was published.

   values(saslib.sasudf_compiler_path)

   You should receive a result similar to one of the following.

   /usr/vac/bin  /* on AIX */
   /usr/bin      /* on Linux */

3. Enter the following command to verify that the SASUDF_DB2PATH global variable was published.

   values(saslib.sasudf_db2path)

   You should receive a result similar to the following.

   /users/db2v10/sqllib

   In this example, /users/db2v10 is the value of db2instancepath that was specified during installation and /users/db2v10/sqllib is also where the SAS_COMPILEUDF function was published.

4. Enter the following command to verify that the SAS_COMPILEUDF and SAS_DELETEUDF functions were published.

   select funcname, implementation from syscat.functions where funcschema='SASLIB'
DB2SET Command Syntax for In-Database Processing

The DB2SET command has many arguments. The syntax for the DB2SET command that is used with the SAS Embedded Process is shown below.

```
DB2SET DB2_SAS_SETTINGS="
ENABLE_SAS_EP:TRUE | FALSE;
LIBRARY_PATH:path
\; COMM_BUFFER_SZ:size>
\; COMM_TIMEOUT:timeout
\; RESTART RETRIES:number-of-tries
\; DIAGLEVEL:level-number>"; DIAGPATH:path>
"
```

Arguments

**ENABLE_SAS_EP:TRUE | FALSE**

specifies whether the SAS Embedded Process is started with the DB2 instance.

Default FALSE

**LIBRARY_PATH:path**

specifies the path from which the SAS Embedded Process library is loaded.

Requirement The path must be fully qualified.

**COMM_BUFFER_SZ:size**

specifies the size in 4K pages of the shared memory buffer that is used for communication sessions between DB2 and SAS.

Default ASLHEAPSZ dbm configuration value

Range 1–32767

Requirement size must be an integer value.

**COMM_TIMEOUT:timeout**

specifies a value in seconds that DB2 uses to determine whether the SAS Embedded Process is non-responsive when DB2 and SAS are exchanging control messages.

Default 600 seconds

Note If the time-out value is exceeded, DB2 forces the SAS Embedded Process to stop in order for it to be re-spawned.
RESTART_RETRIES:number-of-tries
specifies the number of times that DB2 attempts to re-spawn the SAS Embedded Process after DB2 has detected that the SAS Embedded Process has terminated abnormally.

Default: 10
Range: 1–100
Requirement: number-of-tries must be an integer value.

Note: When DB2 detects that the SAS Embedded Process has terminated abnormally, DB2 immediately attempts to re-spawn it. This argument limits the number of times that DB2 attempts to re-spawn the SAS Embedded Process. Once the retry count is exceeded, DB2 waits 15 minutes before trying to re-spawn it again.

DIAGPATH:path
specifies the path that indicates where the SAS Embedded Process diagnostic logs are written.

Default: DIAGPATH dbm configuration value
Requirement: The path must be fully qualified.

DIAGLEVEL:level-number
specifies the minimum severity level of messages that are captured in the SAS Embedded Process diagnostic logs. The levels are defined as follows.

1 SEVERE
2 ERROR
3 WARNING
4 INFORMATIONAL

Default: DIAGLEVEL dbm configuration value
Range: 1–4

Controlling the SAS Embedded Process for DB2

The SAS Embedded Process starts when a query is submitted. The SAS Embedded Process continues to run until it is manually stopped or the database is shut down.

The DB2IDA command is a utility that is installed with the DB2 server to control the SAS Embedded Process. The DB2IDA command enables you to manually stop and restart the SAS Embedded Process without shutting down the database. You might use the DB2IDA command to upgrade or re-install the SAS Embedded Process library or correct an erroneous library path.

Note: DB2IDA requires IBM Fixpack 6 or later.

The DB2IDA command has the following parameters:
-provider sas
  specifies the provider that is targeted by the command. The only provider that is
  supported is "sas".

-start
  starts the SAS Embedded Process on the DB2 instance if the SAS Embedded Process
  is not currently running.

  If the SAS Embedded Process is running, this command has no effect.

  Note: Once the SAS Embedded Process is started, the normal re-spawn logic in DB2
  applies if the SAS Embedded Process is abnormally terminated.

-stop
  stops the SAS Embedded Process if it is safe to do so.

  If the SAS Embedded Process is stopped, this command has no effect.

  If any queries are currently running on the SAS Embedded Process, the db2ida -stop
  command fails and indicates that the SAS Embedded Process is in use and could not
  be stopped.

  Note: DB2 does not attempt to re-spawn the SAS Embedded Process once it has
  been stopped with the db2ida -stop command.

-stopforce
  forces the SAS Embedded Process to shut down regardless of whether there are any
  queries currently running on it.

  If the SAS Embedded Process is stopped, this command has no effect.

  If any queries are currently running on the SAS Embedded Process, those queries
  receive errors.

  Note: DB2 does not attempt to re-spawn the SAS Embedded Process once it has
  been stopped with the db2ida -stopforce command.

Here are some examples of the DB2IDA command:

  db2ida -provider sas -stopforce

  db2ida -provider sas -start

---

**DB2 Permissions**

There are two sets of permissions involved with the in-database software.

- The first set of permissions is needed by the person who publishes the
  SAS_COMPILEUDF and SAS_DELETEUDF functions and creates the
  SAS_UDF_COMPILER_PATH and SAS_UDF_DB2PATH global variables.

  These permissions must be granted before the %INDB2_PUBLISH_COMPILEUDF
  and %INDB2_PUBLISH_DELETEUDF macros are run. Without these permissions,
  running these macros fails.

  The following table summarizes the permissions that are needed by the person who
  publishes the functions and creates the global variables.
### Table 3.1 Permissions Needed to Publish the Functions and Create Global Variables

<table>
<thead>
<tr>
<th>Permission Needed</th>
<th>Authority Required to Grant Permission</th>
<th>Examples</th>
</tr>
</thead>
</table>
| CREATEIN permission for the SASLIB schema in which the SAS_COMPILEUDF and SAS_DELETEUDF functions are published and the SASUDF_COMPILER_PATH and SASUDF_DB2PATH global variables are defined | System Administrator or Database Administrator
  *Note:* If you have SYSADM or DBADM authority or are the DB2 instance owner, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions. | GRANT CREATEIN ON SCHEMA SASLIB TO compiledeletepublisheruserid |
| CREATE_EXTERNAL_ROUTINE permission to the database in which the SAS_COMPILEUDF and SAS_DELETEUDF functions are published | System Administrator or Database Administrator | GRANT CREATE_EXTERNAL_ROUTINE ON DATABASE TO compile-delete-publisher-userid |

- The second set of permissions is needed by the person who publishes the format or scoring model functions. The person who publishes the format or scoring model functions is not necessarily the same person who publishes the SAS_COMPILEUDF and SAS_DELETEUDF functions and creates the SASUDF_COMPILER_PATH and SASUDF_DB2PATH global variables. These permissions are most likely needed by the format publishing or scoring model developer. Without these permissions, the publishing of the format or scoring model functions fails.

  *Note:* Permissions must be granted for every format or scoring model publisher and for each database that the format or scoring model publishing uses. Therefore, you might need to grant these permissions multiple times.

  *Note:* If you are using the SAS Embedded Process to run your scoring functions, only the CREATE TABLE permission is needed.

After the DB2 permissions have been set appropriately, the format or scoring publishing macro should be called to register the formats or scoring model functions. The following table summarizes the permissions that are needed by the person who publishes the format or scoring model functions.
Table 3.2 Permissions Needed to Publish the Format or Scoring Model Functions

<table>
<thead>
<tr>
<th>Permission Needed</th>
<th>Authority Required to Grant Permission</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE permission for functions that have been published.</td>
<td>System Administrator or Database Administrator</td>
<td>GRANT EXECUTE ON FUNCTION SASLIB.* TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>This enables the person who publishes the formats or scoring model functions to execute the SAS_COMPILEUDF and SAS_DELETEUDF functions.</td>
<td></td>
<td>Note: If you have SYSADM or DBADM authority, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions.</td>
</tr>
<tr>
<td>CREATE_EXTERNAL_ROUTINE permission to the database to create format or scoring model functions</td>
<td></td>
<td>GRANT CREATE_EXTERNAL_ROUTINE ON DATABASE TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>CREATE_NOT_FENCED_ROUTINE permission to create format or scoring model functions that are not fenced</td>
<td></td>
<td>GRANT CREATE_NOT_FENCED_ROUTINE ON DATABASE TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>CREATEIN permission for the schema in which the format or scoring model functions are published if the default schema (SASLIB) is not used</td>
<td></td>
<td>GRANT CREATEIN ON SCHEMA scoring-schema TO scoring-or-fmt-publisher-rid</td>
</tr>
<tr>
<td>CREATE TABLE permission to create the model table used in with scoring and the SAS Embedded Process</td>
<td>Person who ran the %INDB2_PUBLISH_COMPILEUDF macro</td>
<td>GRANT CREATETAB TO scoring-publisher-SEP-id</td>
</tr>
<tr>
<td>READ permission to read the SASUDF_COMPILER_PATH and SASUDF_DB2PATH global variables</td>
<td>Person who ran the %INDB2_PUBLISH_COMPILEUDF macro</td>
<td>GRANT READ ON VARIABLE SASLIB.SASUDF_COMPILER_PATH TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>Note: The person who ran the %INDB2_PUBLISH_COMPILEUDF macro has these READ permissions and does not need to grant them to himself or herself again.</td>
<td>Note: For security reasons, only the user who created these variables has the permission to grant READ permission to other users. This is true even for the user with administrator permissions such as the DB2 instance owner.</td>
<td>GRANT READ ON VARIABLE SASLIB.SASUDF_DB2PATH TO scoring-or-fmt-publisher-id</td>
</tr>
</tbody>
</table>

Note: If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, additional permissions are required. For more information, see Chapter 12, “Configuring SAS Model Manager,” on page 115.

Documentation for Using In-Database Processing in DB2

For information about how to publish SAS formats or scoring models, see SAS In-Database Products: User’s Guide.
Part 4

Administrator’s Guide for Greenplum

Chapter 4
Greenplum Administrator’s Guide
Chapter 4
Greenplum Administrator’s Guide

Prerequisites

The in-database deployment package for Greenplum requires the following:

- SAS Foundation and the SAS/ACCESS Interface to Greenplum. This software must be installed before you install and configure the in-database deployment package for Greenplum.

- a specific version of the Greenplum client and server environment and the Greenplum Partner Connector (GPPC) API. For more information, see the SAS Foundation system requirements documentation for your operating environment.

- To publish the utility (SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORUDF, SAS_DEHEXUDF, SAS_EP), format, and scoring model functions, Greenplum requires that you have superuser permissions to create and execute these functions in the SASLIB (or other specified) schema and in the specified database.
For more information, see “Greenplum Permissions” on page 48.

- the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Greenplum to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your Greenplum system. For more information, see SAS Embedded Process: Deployment Guide.

---

## Overview of the In-Database Deployment Package for Greenplum

The in-database deployment package for Greenplum contains the SAS formats library and precompiled binary files for the utility functions. The in-database deployment package is available for installation after you deploy SAS/ACCESS Interface to Greenplum.

The in-database deployment package for Greenplum must be installed and configured before you can perform the following tasks:

- Use the %INDGP_PUBLISH_FORMATS format publishing macro to create or publish the SAS_PUT() function and to create or publish user-defined formats as format functions inside the database.

- Use the %INDGP_PUBLISH_MODEL scoring publishing macro to create scoring files and functions inside the database.

- Use the SAS In-Database Code Accelerator for Greenplum to execute DS2 thread programs in parallel inside the database.

- Run SAS High-Performance Analytics when the analytics cluster is co-located with the Greenplum data appliance or when the analytics cluster is using a parallel connection with a remote Greenplum data appliance. The SAS Embedded Process, which resides on the data appliance, is used to provide high-speed parallel data transfer between the data appliance and the analytics environment where it is processed.

  For more information, see the SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide.

For more information about using the format and scoring publishing macros and the SAS In-Database Code Accelerator for Greenplum, see the SAS In-Database Products: User’s Guide.

The SAS formats library is a run-time library that is installed on your Greenplum system. This installation is done so that the SAS scoring model functions and the SAS_PUT() function created in Greenplum can access the routines within the run-time library. The SAS formats library contains the formats that are supplied by SAS.

The %INDGP_PUBLISH_COMPILEUDF macro registers utility functions in the database. The utility functions are called by the format and scoring publishing macros: %INDGP_PUBLISH_FORMATS and %INDGP_PUBLISH_MODEL. You must run the %INDGP_PUBLISH_COMPILEUDF macro before you run the format and scoring publishing macros.

The %INDGP_PUBLISH_COMPILEUDF_EP macro defines the SAS_EP table functions to the Greenplum database. You use the SAS_EP table function to produce scoring models after you run the %INDGP_PUBLISH_MODEL macro to create the SAS scoring files and publish them to the scoring model table. The SAS Embedded
Process accesses the SAS scoring files when a scoring operation is performed. You also use the SAS_EP table function for other SAS software that requires it, such as SAS High-Performance Analytics.

Greenplum Installation and Configuration

Greenplum Installation and Configuration Steps

1. If you are upgrading from a previous release, follow the instructions in “Upgrading from a Previous Version” on page 35 before installing the in-database deployment package.

2. Install the SAS formats library and the binary files.
   For more information, see “Installing the SAS Formats Library and Binary Files” on page 37.

3. Run the %INDGP_PUBLISH_COMPILEUDF macro if you want to publish formats or use scoring functions to run a scoring model. Run the %INDGP_PUBLISH_COMPILEUDF_EP macro if you want to use the SAS Embedded Process to run a scoring model or other SAS software that requires it.
   For more information, see “Running the %INDGP_PUBLISH_COMPILEUDF Macro” on page 38 or “Running the %INDGP_PUBLISH_COMPILEUDF_EP Macro” on page 42.

4. If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, perform the additional configuration tasks provided in Chapter 12, “Configuring SAS Model Manager,” on page 115.

Note: If you plant to use the SAS High-Performance Analytics environment, there are additional steps to be performed after you install the SAS Embedded Process. For more information, see SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide.

Upgrading from a Previous Version

Upgrading the 9.3 SAS Formats Library and Binary Files
To upgrade from the SAS 9.3 version, follow these steps:

1. Delete the full-path-to-pkglibdir/SAS directory that contains the SAS formats library and binary files. For SAS 9.3, this directory also contains the SAS Embedded Process (SAS 9.3).

Note: You can use the following command to determine the full-path-to-pkglibdir directory.

```
pkg_config --pkglibdir
```

If you did not perform the Greenplum install, you cannot run the `pkg_config --pkglibdir` command. The `pkg_config --pkglibdir` command must be run by the person who performed the Greenplum installation.

CAUTION:
If you delete the SAS directory, all the scoring models that you published using scoring functions and all user-defined formats that you published are
If you previously published scoring models using scoring functions or if you previously published user-defined formats, you must republish your scoring models and formats. If you used the SAS Embedded Process to publish scoring models, the scoring models are not deleted.

It is a best practice to delete the SAS directory when you upgrade from a previous version. Doing so ensures that you get the latest version of the SAS Formats Library and binary files.

2. Continue the installation instructions in “Installing the SAS Formats Library and Binary Files” on page 37.

Upgrading the 9.4 SAS Formats Library and Binary Files

To upgrade from the SAS 9.4 version, follow these steps. If you upgrade or install the SAS Formats Library and binary files in this manner, you do not delete any scoring models or formats that were previously published.

1. Log on to the Greenplum master node as a superuser.

2. Run the UninstallSASEPFiles.sh file.

   ```bash
   ./UninstallSASEPFiles.sh
   ```

   This script stops the SAS Embedded Process on each database host node. The script deletes the `/SAS/SASTKInDatabaseServerForGreenplum` directory and all its contents from each database host node.

   Prior to SAS 9.4M4 (November 2016), the UninstallSASEPFiles.sh file is in the `path_to_sh_file` directory where you copied the tkindbsrv self-extracting archive file.

   Starting with SAS 9.4M4 (November 2016), the UninstallSASEPFiles.sh file is in the `path_to_sh_file/SASEPHome/bin` directory.

   **CAUTION:**

   The timing option must be off for the UninstallSASEPFiles.sh scripts to work. Put `timing off` in your `.psqlrc` file before running this script.

3. Move to the directory where the SAS Formats Library is installed.

   The directory path is `full-path-to-pkglibdir/SAS/`.

   **Note:** You can use the following command to determine the `full-path-to-pkglibdir` directory.

   ```bash
   pg_config --pkglibdir
   ```

   If you did not perform the Greenplum install, you cannot run the `pg_config --pkglibdir` command. The `pg_config --pkglibdir` command must be run by the person who performed the Greenplum install.

4. Delete the libjazxfbrs.so and sas_compileudf.so files.

5. In addition to deleting the libjazxfbrs.so and sas_compileudf.so files on the master node, you must log on to each host node and delete the files on these nodes.

6. Continue the installation instructions in “Installing the SAS Formats Library and Binary Files” on page 37.
Installing the SAS Formats Library and Binary Files

The SAS formats library and the binary files for the publishing macros are contained in a self-extracting archive file. The self-extracting archive file is located in the

YourSASHome/SASFormatsLibraryforGreenplum/3.1/
GreenplumonLinux64/ directory.

To move and unpack the self-extracting archive file, follow these steps:

1. Using a method of your choice, transfer the accelgplmfmt-3.1-n_lax.sh file to your Greenplum master node. \( n \) is a number that indicates the latest version of the file. If this is the initial installation, \( n \) has a value of 1. Each time you re-install or upgrade, \( n \) is incremented by 1.

   The file does not have to be downloaded to a specific location. However, you should note where the file is downloaded so that it can be executed at a later time.

2. After the accelgplmfmt-3.1-n_lax.sh has been transferred, log on to the Greenplum master node as a superuser.

3. Move to the directory where the self-extracting archive file was downloaded.

4. Use the following command at the UNIX prompt to unpack the self-extracting archive file:

   `./accelgplmfmt-3.1-n_lax.sh`

   Note: If you receive a “permissions denied” message, check the permissions on the accelgplmfmt-3.1-n_lax.sh file. This file must have EXECUTE permissions to run.

After the script runs and the files are unpacked, the content of the target directories should look similar to these where `path_to_sh_file` is the location to which you copied the self-extracting archive file.

```
/path_to_sh_file/SAS/SASFormatsLibraryForGreenplum/3.1-1/bin/
   InstallAccelGplmFmt.sh

/path_to_sh_file/SAS/SASFormatsLibraryForGreenplum/3.1-1/bin/
   CopySASFiles.sh

/path_to_sh_file/SAS/SASFormatsLibraryForGreenplum/3.1-1/lib/
   SAS_CompileUDF.so

/path_to_sh_file/SAS/SASFormatsLibraryForGreenplum/3.1-1/lib/
   libjazxfbrs.so
```

5. Use the following command to place the files in Greenplum:

   `./path_to_sh_file/SAS/SASFormatsLibraryForGreenplum/3.1-1/bin/
   CopySASFiles.sh`

   **CAUTION:**
   
   The timing option must be off for the CopySASFiles.sh script to work. Put `\( \text{timing off} \)` in your .psqlrc file before running this script.

This command replaces all previous versions of the libjazxfbrs.so file.

All the SAS object files are stored under `full-path-to-pkglibdir/SAS`. The files are copied to the master node and each of the segment nodes.

**Note:** You can use the following command to determine the `full-path-to-pkglibdir` directory:

```
pkg_config --pkglibdir
```
If you did not perform the Greenplum install, you cannot run the `pg_config --pkglibdir` command. The `pg_config --pkglibdir` command must be run by the person who performed the Greenplum install.

**Note:** If you add new nodes at a later date, you must copy all the binary files to the new nodes. For more information, see Step 6.

6. (Optional) If you add new nodes to the Greenplum master node after the initial installation of the SAS formats library and publishing macro, you must copy all the binaries in the `full-path-to-pkglibdir/SAS` directory to the new nodes using a method of your choice such as `scp /SAS`. The binary files include `SAS_CompileUDF.so`, `libjazxfbrs.so`, and the binary files for the already published functions.

### Running the %INDGP_PUBLISH_COMPILEUDF Macro

**Overview of the %INDGP_PUBLISH_COMPILEUDF Macro**

Use the `%INDGP_PUBLISH_COMPILEUDF` macro if you want to publish formats or use scoring functions to run scoring models.

**Note:** Use the `%INDGP_PUBLISH_COMPILEUDF_EP` macro if you need to use the SAS Embedded Process. For more information, see “Running the %INDGP_PUBLISH_COMPILEUDF_EP Macro” on page 42.

The `%INDGP_PUBLISH_COMPILEUDF` macro publishes the following functions to the SASLIB schema in a Greenplum database:

- **SAS_COMPILEUDF function**

  This function facilitates the `%INDGP_PUBLISH_FORMATS` format publishing macro and the `%INDGP_PUBLISH_MODEL` scoring publishing macro. The SAS_COMPILEUDF function performs the following tasks:
  - compiles the format and scoring model source files into object files. This compilation occurs through the SQL interface using an appropriate compiler for the system.
  - links with the SAS formats library.
  - copies the object files to the `full-path-to-pkglibdir/SAS` directory. All the SAS object files are stored under `full-path-to-pkglibdir/SAS`.

  **Note:** You can use the following command to determine the `full-path-to-pkglibdir` directory:

  ```
  pg_config --pkglibdir
  ```

  If you did not perform the Greenplum install, you cannot run the `pg_config --pkglibdir` command. The `pg_config --pkglibdir` command must be run by the person who performed the Greenplum install.

- **three utility functions. These are used when the format and scoring publishing macros transfer source files from the client to the host:**
  - **SAS_COPYUDF function**

    This function copies the shared libraries to the `full-path-to-pkglibdir/SAS` path on the whole database array including the master and all segments.
  - **SAS_DIRECTORYUDF function**
This function creates and removes a temporary directory that holds the source files on the server.

- **SAS_DEHEXUDF function**

  This function converts the files from hexadecimal back to text after the files are exported on the host.

You have to run the `%INDGP_PUBLISH_COMPILEUDF` macro only one time in each database.

*Note:* The SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions must be published before you run the `%INDGP_PUBLISH_FORMATS` or the `%INDGP_PUBLISH_MODEL` macro. Otherwise, these macros fail.

*Note:* To publish the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions, you must have superuser permissions to create and execute these functions in the SASLIB schema and in the specified database.

**%INDGP_PUBLISH_COMPILEUDF Macro Run Process**

To run the `%INDGP_PUBLISH_COMPILEUDF` macro, follow these steps:

*Note:* To publish the SAS_COMPILEUDF function, you must have superuser permissions to create and execute this function in the SASLIB schema and in the specified database.

1. Create a SASLIB schema in the database where the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions are published.

   You must use “SASLIB” as the schema name for Greenplum in-database processing to work correctly.

   You specify that database in the DATABASE argument of the `%INDGP_PUBLISH_COMPILEUDF` macro. For more information, see “%INDGP_PUBLISH_COMPILEUDF Macro Syntax” on page 41.

   The SASLIB schema contains the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions.

2. Start SAS 9.4 and submit the following command in the Enhanced Editor or Program Editor:

   ```
   %let indconn = user=youruserid password=yourpwd dsn=yourdsn;
   /* You can use server=yourserver database=yourdb instead of dsn=yourdsn */
   
   For more information, see the “INDCONN Macro Variable” on page 40.
   ```

3. Run the `%INDGP_PUBLISH_COMPILEUDF` macro.

   For more information, see “%INDGP_PUBLISH_COMPILEUDF Macro Syntax” on page 41.

You can verify that the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions have been published successfully. For more information, see “Validating the Publishing of the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF Functions” on page 45.
### INDCONN Macro Variable

The INDCONN macro variable provides the credentials to make a connection to Greenplum. You must specify the user, password, and either the DSN or server and database information to access the machine on which you have installed the Greenplum database. You must assign the INDCONN macro variable before the %INDGP_PUBLISH_COMPILEUDF macro is invoked.

The value of the INDCONN macro variable for the %INDGP_PUBLISH_COMPILEUDF macro has one of these formats:

- `USER=userid`  
- `PASSWORD=password`  
- `DSN=dsnname`  
- `<PORT=port-number>`

- `USER=userid`  
- `PASSWORD=password`  
- `SERVER=server`  
- `DATABASE=database`  
- `<PORT=port-number>`

- `USER=userid | 'userid'`

  specifies the Greenplum user name (also called the user ID) that is used to connect to the database. If the user name contains spaces or nonalphanumeric characters, enclose the user name in quotation marks.

- `PASSWORD=password | 'password'`

  specifies the password that is associated with your Greenplum user ID. If the password contains spaces or nonalphabetic characters, enclose the password in quotation marks.

  **Tip** Use only `PASSWORD=`, `PASS=`, or `PW=` for the password argument. `PWD=` is not supported and causes an error.

- `DSN=datasource | 'datasource'`

  specifies the configured Greenplum ODBC data source to which you want to connect. If the DSN name contains spaces or nonalphabetic characters, enclose the DSN name in quotation marks.

  **Requirement** You must specify either the `DSN=` argument or the `SERVER=` and `DATABASE=` arguments in the INDCONN macro variable.

- `SERVER=server | 'server'`

  specifies the Greenplum server name or the IP address of the server host. If the server name contains spaces or nonalphanumeric characters, enclose the server name in quotation marks.

  **Requirement** You must specify either the `DSN=` argument or the `SERVER=` and `DATABASE=` arguments in the INDCONN macro variable.

- `DATABASE=database | 'database'`

  specifies the Greenplum database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, enclose the database name in quotation marks.

  **Requirement** You must specify either the `DSN=` argument or the `SERVER=` and `DATABASE=` arguments in the INDCONN macro variable.

- `PORT=port-number | 'port-number'`

  specifies the `psql` port number.

  **Default** 5432
Requirement  The server-side installer uses psql, and psql default port is 5432. If you want to use another port, you must have the UNIX or database administrator change the psql port.

*Note:* The SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions are published to the SASLIB schema in the specified database. The SASLIB schema must be created before publishing the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions.

%INDGP_PUBLISH_COMPILEUDF Macro Syntax

%INDGP_PUBLISH_COMPILEUDF

(OBJPATH=full-path-to-pkglibdir/SAS
 <, DATABASE=database-name>
 <, ACTION=CREATE | REPLACE | DROP>
 <, OUTDIR=diagnostic-output-directory>
);

**Arguments**

**OBJPATH=full-path-to-pkglibdir/SAS**  
specifies the parent directory where all the object files are stored.

**Tip**  
The *full-path-to-pkglibdir* directory was created during installation of the InstallAccelGplmFmt.sh self-extracting archive file. You can use the following command to determine the *full-path-to-pkglibdir* directory:

```
pg_config --pkglibdir
```

If you did not perform the Greenplum install, you cannot run the `pg_config --pkglibdir` command. The `pg_config --pkglibdir` command must be run by the person who performed the Greenplum install.

**DATABASE=database-name**  
specifies the name of a Greenplum database to which the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions are published.

**Restriction**  
If you specify DSN= in the INDCONN macro variable, do not use the DATABASE argument.

**ACTION=CREATE | REPLACE | DROP**  
specifies that the macro performs one of the following actions:

**CREATE**  
creates a new SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF function.

**REPLACE**  
overwrites the current SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions, if a function by the same name is already registered, or creates a new SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF function if one is not registered.

**Requirement**  
If you are upgrading the SAS Formats Library, run the %INDGP_PUBLISH_COMPILEUDF macro with ACTION=REPLACE. The CopySASFiles.sh install script replaces existing versions of most files. However, you need to
replace the existing SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions after you run the CopySASFiles.sh install script. For more information, see “Upgrading from a Previous Version” on page 35 and “Installing the SAS Formats Library and Binary Files ” on page 37.

**DROP**
causes the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions to be dropped from the Greenplum database.

**Default** CREATE

**Tip** If the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions were published previously and you specify ACTION=CREATE, you receive warning messages that the functions already exist and you are prompted to use REPLACE. If the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions were published previously and you specify ACTION=REPLACE, no warnings are issued.

**OUTDIR=** diagnostic-output-directory
specifies a directory that contains diagnostic files.

**Tip** Files that are produced include an event log that contains detailed information about the success or failure of the publishing process.

### Running the %INDGP_PUBLISH_COMPILEUDF_EP Macro

**Overview of the %INDGP_PUBLISH_COMPILEUDF_EP Macro**

Use the %INDGP_PUBLISH_COMPILEUDF_EP macro if you want to use the SAS Embedded Process to run scoring models or other SAS software that requires it.

**Note:** Use the %INDGP_PUBLISH_COMPILEUDF macro if you want to use scoring functions to run scoring models. For more information, see “Running the %INDGP_PUBLISH_COMPILEUDF Macro” on page 38.

The %INDGP_PUBLISH_COMPILEUDF_EP macro registers the SAS_EP table functions in the database.

You have to run the %INDGP_PUBLISH_COMPILEUDF_EP macro only one time in each database where scoring models are published.

The %INDGP_PUBLISH_COMPILEUDF_EP macro must be run before you use the SAS_EP function in an SQL query.

**Note:** To publish the SAS_EP function, you must have superuser permissions to create and execute this function in the specified schema and database.

### %INDGP_PUBLISH_COMPILEUDF_EP Macro Run Process

To run the %INDGP_PUBLISH_COMPILEUDF_EP macro, follow these steps:

**Note:** To publish the SAS_EP function, you must have superuser permissions to create and execute this function in the specified schema and database.

1. Create a schema in the database where the SAS_EP function is published.
Note: You must publish the SAS_EP function to a schema that is in your schema search path.

You specify the schema and database in the INDCONN macro variable. For more information, see “INDCONN Macro Variable” on page 43.

2. Start SAS 9.4 and submit the following command in the Enhanced Editor or Program Editor:

   ```
   %let indconn = user=youruserid password=yourpwd dsn=yourdsn <schema=yourschema>;
   /* You can use server=yourserver database=yourdb instead of dsn=yourdsn */
   
   For more information, see the “INDCONN Macro Variable” on page 43.
   
3. Run the `%INDGP_PUBLISH_COMPILEUDF_EP` macro. For more information, see “%INDGP_PUBLISH_COMPILEUDF_EP Macro Syntax” on page 44.

   You can verify that the SAS_EP function has been published successfully. For more information, see “Validating the Publishing of the SAS_EP Function” on page 46.

### INDCONN Macro Variable

The INDCONN macro variable provides the credentials to make a connection to Greenplum. You must specify the user, password, and either the DSN or server and database information to access the machine on which you have installed the Greenplum database. You must assign the INDCONN macro variable before the `%INDGP_PUBLISH_COMPILEUDF_EP` macro is invoked.

The value of the INDCONN macro variable for the `%INDGP_PUBLISH_COMPILEUDF_EP` macro has one of these formats:

- USER=userid PASSWORD=password DSN=dsnname
  `<SCHEMA=schema> <PORT=port-number>`

- USER=userid PASSWORD=password SERVER=server
  DATABASE=database `<SCHEMA=schema>`
  `<PORT=port-number>`

- USER=userid | 'userid'
  specifies the Greenplum user name (also called the user ID) that is used to connect to the database. If the user name contains spaces or nonalphanumeric characters, enclose the user name in quotation marks.

- PASSWORD=password | 'password'
  specifies the password that is associated with your Greenplum user ID. If the password contains spaces or nonalphabetic characters, enclose the password in quotation marks.

  **Tip** Use only PASSWORD=, PASS=, or PW= for the password argument. PWD= is not supported and causes an error.

- DSN=datasource | 'datasource'
  specifies the configured Greenplum ODBC data source to which you want to connect. If the DSN name contains spaces or nonalphabetic characters, enclose the DSN name in quotation marks.

**Requirement** You must specify either the DSN= argument or the SERVER= and DATABASE= arguments in the INDCONN macro variable.
**SERVER**='server' | 'server'
specifies the Greenplum server name or the IP address of the server host. If the server name contains spaces or nonalphanumeric characters, enclose the server name in quotation marks.

**Requirement**
You must specify either the **DSN**= argument or the **SERVER**= and **DATABASE**= arguments in the INDCONN macro variable.

**DATABASE**='database' | 'database'
specifies the Greenplum database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, enclose the database name in quotation marks.

**Requirement**
You must specify either the **DSN**= argument or the **SERVER**= and **DATABASE**= arguments in the INDCONN macro variable.

**SCHEMA**='schema' | 'schema'
specifies the name of the schema where the SAS_EP function is defined.

**Default**
SASLIB

**Requirements**
You must create the schema in the database before you run the %INDGP_PUBLISH_COMPILEUDF_EP macro.

You must publish the SAS_EP function to a schema that is in your schema search path.

**PORT**='port-number' | 'port-number'
specifies the psql port number.

**Default**
5432

**Requirement**
The server-side installer uses psql, and psql default port is 5432. If you want to use another port, you must have the UNIX or database administrator change the psql port.

### %INDGP_PUBLISH_COMPILEUDF_EP Macro Syntax

%INDGP_PUBLISH_COMPILEUDF_EP

\(<OBJPATH=full-path-to-pkglibdir/SAS>\)

\(<, DATABASE=database-name>\)

\(<, ACTION=CREATE | REPLACE | DROP>\)

\(<, OUTDIR=diagnostic-output-directory>\)

);

**Arguments**

**OBJPATH=full-path-to-pkglibdir/SAS**
specifies the parent directory where all the object files are stored.

**Tip**
The **full-path-to-pkglibdir** directory was created during installation of the InstallSASEP.sh self-extracting archive file. You can use the following command to determine the **full-path-to-pkglibdir** directory:

\(pg\_config --pkglibdir\)

If you did not perform the Greenplum install, you cannot run the \(pg\_config --pkglibdir\) command. The \(pg\_config --pkglibdir\) command must be run by the person who performed the Greenplum install.
**DATABASE=** *database-name*

specifies the name of a Greenplum database where the SAS_EP function is defined.

**Restriction**  
If you specify DSN= in the INDCONN macro variable, do not use the DATABASE argument.

**ACTION=CREATE | REPLACE | DROP**

specifies that the macro performs one of the following actions:

**CREATE**

creates a new SAS_EP function.

**REPLACE**

overwrites the current SAS_EP function, if a function by the same name is already registered, or creates a new SAS_EP function if one is not registered.

**Requirement**  
If you are upgrading the SAS Embedded Process, run the `%INDGP_PUBLISH_COMPILEUDF_EP` macro with ACTION=REPLACE. The InstallSASEPFiles.sh install script replaces existing versions of most files. However, you need to replace the existing SAS_EP function after you run the InstallSASEPFiles.sh install script. For more information, see “Upgrading from a Previous Version” on page 35 and “Installing the SAS Formats Library and Binary Files” on page 37.

**DROP**

causes the SAS_EP function to be dropped from the Greenplum database.

**Default** CREATE

**Tip**  
If the SAS_EP function was defined previously and you specify ACTION=CREATE, you receive warning messages that the functions already exist and you are prompted to use REPLACE. If the SAS_EP function was defined previously and you specify ACTION=REPLACE, no warnings are issued.

**OUTDIR=** *diagnostic-output-directory*

specifies a directory that contains diagnostic files.

**Tip** Files that are produced include an event log that contains detailed information about the success or failure of the publishing process.

---

**Validation of Publishing Functions**

**Validating the Publishing of the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF Functions**

To validate that the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF functions are registered properly under the SASLIB schema in the specified database, follow these steps.

1. Use `psql` to connect to the database.
psql -d databasename

You should receive the following prompt.

databasename=#

2. At the prompt, enter the following command.

```sql
select prosrc from pg_proc f, pg_namespace s where f.pronamespace=s.oid
  and upper(s.nspname)='SASLIB';
```

You should receive a result similar to the following:

- SAS_CompileUDF
- SAS_CopyUDF
- SAS_DirectoryUDF
- SAS_DehexUDF

Validating the Publishing of the SAS_EP Function

To validate that the SAS_EP function is registered properly under the specified schema in the specified database, follow these steps.

1. Use psql to connect to the database.

```sql
psql -d databasename
```

You should receive the following prompt.

```
databasename=
```

2. At the prompt, enter the following command.

```sql
select proname, prosrc, probin, nspname FROM pg_catalog.pg_namespace n
  JOIN pg_catalog.pg_proc p ON pronamespace = n.oid
WHERE proname = 'sas_ep'
  or proname = 'sas_ep_describe';
```

You should receive a result similar to the following:

<table>
<thead>
<tr>
<th>proname</th>
<th>prosrc</th>
<th>probin</th>
</tr>
</thead>
<tbody>
<tr>
<td>sas_ep_describe</td>
<td>SAS_EP_Describe</td>
<td>$libdir/SASEPHome/sasexe/sasep_tablefunc.so</td>
</tr>
<tr>
<td></td>
<td>&lt;schema&gt;</td>
<td></td>
</tr>
<tr>
<td>sas_ep</td>
<td>SAS_EP</td>
<td>$libdir/SASEPHome/sasexe/sasep_tablefunc.so</td>
</tr>
<tr>
<td></td>
<td>&lt;schema&gt;</td>
<td></td>
</tr>
</tbody>
</table>

3. Exit psql.

```
\q
```

Controlling the SAS Embedded Process

The SAS Embedded Process starts when a query is submitted using the SAS_EP function. It continues to run until it is manually stopped or the database is shut down.

*Note:* Starting and stopping the SAS Embedded Process has implications for all scoring model publishers.
Note: Manually starting and stopping the SAS Embedded Process requires superuser permissions and must be done from the Greenplum master node.

When the SAS Embedded Process is installed, the ShutdownSASEP.sh and StartupSASEP.sh scripts are installed in the following directory. For more information about these files, see “Upgrading or Uninstalling the SAS Embedded Process for Greenplum” in SAS Embedded Process: Deployment Guide.

/opt/sas/ep/home/bin/

CAUTION: The timing option must be off for any of the .sh scripts to work. Put \texttt{timing off} in your .psqlrc file before running these scripts.

Use the following command to shut down the SAS Embedded Process:

/opt/sas/ep/home/bin/ShutdownSASEP.sh <-quiet>

When invoked from the master node, ShutdownSASEP.sh shuts down the SAS Embedded Process on each database node. The \texttt{-verbose} option is on by default and provides a status of the shutdown operations as they occur. You can specify the \texttt{-quiet} option to suppress messages. This script should not be used as part of the normal operation. It is designed to be used to shut down the SAS Embedded Process prior to a database upgrade or re-install.

Use the following command to start the SAS Embedded Process:

/opt/sas/ep/home/bin/StartupSASEP.sh <-quiet>

When invoked from the master node, StartupSASEP.sh manually starts the SAS Embedded Process on each database node. The \texttt{-verbose} option is on by default and provides a status of the installation as it occurs. You can specify the \texttt{-quiet} option to suppress messages. This script should not be used as part of the normal operation. It is designed to be used to manually start the SAS Embedded Process and only after consultation with SAS Technical Support.

Semaphore Requirements When Using the SAS Embedded Process for Greenplum

Each time a query using a SAS_EP table function is invoked to execute a score, it requests a set of semaphore arrays (sometimes referred to as semaphore "sets") from the operating system. The SAS Embedded Process releases the semaphore arrays back to the operating system after scoring is complete.

The number of semaphore arrays required for a given SAS Embedded Process execution is a function of the number of Greenplum database segments that are engaged for the query. The Greenplum system determines the number of segments to engage as part of its query plan based on a number of factors, including the data distribution across the appliance.

The SAS Embedded Process requires five semaphore arrays per database segment that is engaged. The maximum number of semaphore arrays required per database host node per SAS Embedded Process execution can be determined by the following formula:

\[
\text{maximum\_number\_semaphore\_arrays} = 5 \times \text{number\_database\_segments}
\]

Here is an example. On a full-rack Greenplum appliance configured with 16 host nodes and six database segment servers per node, a maximum of 30 (5 * 6) semaphore arrays are required on each host node per concurrent SAS Embedded Process execution of a
score. If the requirement is to support the concurrent execution by the SAS Embedded Process of 10 scores, then the SAS Embedded Process requires a maximum of 300 (5* 6 * 10) semaphore arrays on each host node.

SAS recommends that you configure the semaphore array limit on the Greenplum appliance to support twice the limit that is configured by default on the appliance. For example, if the default limit is 2048, double the default limit to 4096.

*Note:* The semaphore limit discussed here is the limit on the number of "semaphore arrays", where each semaphore array is allocated with an application-specified number of semaphores. For the SAS Embedded Process, the limit on the number of semaphore arrays is distinct from the limit on the "maximum number of semaphores system wide". The SAS Embedded Process requests semaphore arrays with two or fewer semaphores in each array. The limit on the maximum semaphores system wide should not need to be increased. The Linux `ipcs -sl` command output shows the typical default semaphore-related limits set on a Greenplum appliance:

```
------- Semaphore Limits -------
max number of arrays = 2048
max semaphores per array = 250
max semaphores system wide = 512000
max ops per semop call = 100
semaphore max value = 32767
```

---

### Greenplum Permissions

To publish the utility (SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, SAS_DEHEXUDF, SAS_EP), format, and scoring model functions, Greenplum requires that you have superuser permissions to create and execute these functions in the SASLIB (or other specified) schema and in the specified database.

In addition to Greenplum superuser permissions, you must have CREATE TABLE permission to create a model table when using the SAS Embedded Process.

If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, additional permissions are required. For more information, see Chapter 12, “Configuring SAS Model Manager,” on page 115.

---

### Documentation for Using In-Database Processing in Greenplum

For information about how to publish SAS formats and scoring models or how to use the SAS In-Database Code Accelerator, see the *SAS In-Database Products: User’s Guide.*
Part 5

Administrator’s Guide for Hadoop

Chapter 5
Hadoop Administrator’s Guide .......................... 51

Chapter 6
Additional Configuration for the SAS Embedded Process ........ 55
Prerequisites for Installing the In-Database Deployment Package for Hadoop

In-database processing for Hadoop requires the following:

- SAS Foundation and the SAS/ACCESS Interface to Hadoop. This software must be installed before you configure in-database processing for Hadoop.

- A specific version of the Hadoop client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

- The SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within MapReduce or Spark to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your Hadoop cluster. For more information, see *SAS Embedded Process: Deployment Guide*.

- Required Hadoop JAR and configuration files are available to the SAS client machine.

Depending on your SAS software, there are several ways these JAR and configuration files are gathered. Gathering the JAR and configuration files is a one-time process (unless you are updating your cluster or changing Hadoop vendors). If you have already gathered the Hadoop JAR and configuration files for another SAS component, you do not need to do it again.

For more information on obtaining the JAR and configuration files, see the following documentation, depending on your SAS software:

- *SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS*
- *SAS Data Loader for Hadoop: Installation and Configuration Guide*
Overview of the SAS Embedded Process for Hadoop

The SAS Embedded Process for Hadoop must be installed and configured on your Hadoop cluster before you can perform the following tasks:

- Run a scoring model in Hadoop Distributed File System (HDFS) using the SAS Scoring Accelerator for Hadoop.
  
  For more information about using the scoring publishing macros, see the *SAS In-Database Products: User’s Guide*.

- Run DATA step scoring programs in Hadoop.
  
  For more information, see the *SAS In-Database Products: User’s Guide*.

- Run DS2 threaded programs in Hadoop using the SAS In-Database Code Accelerator for Hadoop.
  
  For more information, see the *SAS In-Database Products: User’s Guide*.

- Perform data quality operations in Hadoop, transform data in Hadoop, and extract transformed data out of Hadoop for analysis in SAS using the SAS Data Loader for Hadoop.
  
  For more information, see *SAS Data Loader for Hadoop: User’s Guide*.

- Deploy and score text analytic models in Hadoop using SAS Contextual Analysis In-Database Scoring in Hadoop.
  
  For more information, see *SAS Contextual Analysis In-Database Scoring in Hadoop: User’s Guide*.

- Read and write data to HDFS in parallel for SAS High-Performance Analytics.

  *Note:* For deployments that use SAS High-Performance Deployment of Hadoop for the co-located data provider, and access SASHDAT tables exclusively, SAS/ACCESS and SAS Embedded Process are not needed.

  *Note:* If you are installing the SAS High-Performance Analytics environment, you must perform additional steps after you install the SAS Embedded Process. For
more information, see *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.

---

## Hadoop Configuration

All configuration that is required for in-database processing with Hadoop should be complete after you install and configure SAS/ACCESS Interface to Hadoop and the SAS Embedded Process for Hadoop.

The following table provides a checklist.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Where to Go for Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS/ACCESS</td>
<td>Deploy and configure SAS/ACCESS Interface to Hadoop</td>
<td><em>SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS</em></td>
</tr>
<tr>
<td>Hadoop JAR files</td>
<td>Make required Hadoop JAR and configuration files available to the SAS client machine</td>
<td><em>SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS</em></td>
</tr>
</tbody>
</table>
| SAS Embedded Process | Deploy and configure the SAS Embedded Process | • *SAS Embedded Process: Deployment Guide*  
• Chapter 6, “Additional Configuration for the SAS Embedded Process,” on page 55 |
| (Optional) SAS Model Manager | Configure SAS Model Manager | Chapter 12, “Configuring SAS Model Manager,” on page 115 |

---

## Hadoop Permissions

The installation of the SAS Embedded Process for Hadoop involves writing a configuration file to HDFS and deploying files on all data nodes. These tasks require the following permissions:

- Writing the configuration file requires Write permission to HDFS.
  - The HDFS user home directory, `/user/user-account`, must exist. If you deploy the SAS Embedded Process, this user account is used when you manually deploy the SAS Embedded Process section.
- Deploying files across all nodes requires passwordless SSH from the master node to the slave nodes.
Documentation for Using In-Database Processing in Hadoop

For information about using in-database processing in Hadoop, see the following publications:

• FILENAME Statement, Hadoop Access Method in SAS DATA Step Statements: Reference
• High-performance procedures in various SAS publications
• PROC HADOOP and PROC HDMD in Base SAS Procedures Guide
• SAS/ACCESS Interface to Hadoop in SAS/ACCESS for Relational Databases: Reference
• SAS Contextual Analysis In-Database Scoring in Hadoop: User's Guide
• SAS Data Integration Studio: User's Guide
• SAS Data Loader for Hadoop: User's Guide
• SAS DS2 Programmer's Guide
• SAS In-Database Products: User's Guide
• SAS Intelligence Platform: Data Administration Guide
• SAS Model Managers: User's Guide
Chapter 6
Additional Configuration for the SAS Embedded Process

Overview of Additional Configuration Tasks

After you have installed the SAS Embedded Process either manually or by deploying the parcel or stack, the following additional configuration tasks must be performed:

- “Additional Configuration Needed to Use HCatalog File Formats” on page 56.
- “Adding the YARN Application CLASSPATH for MapR” on page 57.
- “Change the Trace Level” on page 57.
- “Adjusting the SAS Embedded Process Performance” on page 58.
- “Specify the Amount of Memory That the SAS Embedded Process Uses” on page 60.
- “Additional Configuration for All Hadoop Distributions – SAS 9.4” on page 61.
- “Additional Configuration for MapR Version 6.0” on page 61.
- “Additional Configuration to Access Amazon S3 Object Storage” on page 61.
- “Additional Configuration When Running Spark on SAS Viya 3.4” on page 63.
Additional Configuration Needed to Use HCatalog File Formats

Overview of HCatalog File Types

HCatalog is a table management layer that presents a relational view of data in the HDFS to applications within Hadoop. With HCatalog, data structures that are registered in the Hive metastore, including SAS data, can be accessed through Hadoop code. HCatalog is part of Apache Hive.

The SAS Embedded Process for Hadoop uses HCatalog to process complex, non-delimited file formats.

Prerequisites for HCatalog Support

If you plan to access complex, non-delimited file types such as Avro or Parquet, the following conditions must be met:

- The Hive JAR files must be on all of the clusters nodes.
- Hive and HCatalog must be installed on all nodes of the cluster.

SAS Client Configuration

Note: If you used the Hadoop tracer script to install the Hadoop JAR files, these configuration tasks are not necessary. SAS client configuration was completed using the script.

The following additional configuration tasks must be performed:

- The hive-site.xml configuration file must be in the SAS_HADOOP_CONFIG_PATH environment variable (SAS 9.4) or the hadoopConfigDir path (SAS Viya).
- The following Hive or HCatalog JAR files must be in the SAS_HADOOP_JAR_PATH (SAS 9.4) or the hadoopJarPath (SAS Viya)

  hive-hcatalog-core-*.jar
  hive-webhcat-java-client-*.jar

For more information about the hadoopConfigDir and the hadoopJarPath paths, see the CASLIB statement in SAS Viya Cloud Analytic Services: Language Reference.

For more information about the SAS_HADOOP_JAR_PATH and SAS_HADOOP_CONFIG_PATH environment variables, see the SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS.

SAS Server-Side Configuration

The SAS Embedded Process installation automatically sets the HCatalog CLASSPATH in the ep-config.xml file. Otherwise, you must manually append the HCatalog CLASSPATH to the MapReduce configuration property, mapreduce.application.classpath, in the mapred-site.xml file on the client side.

Here is an example of an HCatalog CLASSPATH for a Cloudera distribution.
Here is an example of an HCatalog CLASSPATH for a Hortonworks distribution.

```
CDH-version*/bin/../lib/hive/lib/*,
CDH-version*/lib/hive-hcatalog/libexec/.../share/hcatalog/*
```

Adding the YARN Application CLASSPATH for MapR

Two configuration properties specify the YARN application CLASSPATH: yarn.application.classpath and MapReduce.application.classpath. If you do not specify the YARN application CLASSPATH, MapR uses the default CLASSPATH. However, if you specify the MapReduce application CLASSPATH, the YARN application CLASSPATH is ignored. The SAS Embedded Process for Hadoop requires both the YARN application CLASSPATH and the MapReduce application CLASSPATH.

To ensure that the YARN application CLASSPATH exists, you must manually add the YARN application CLASSPATH to the yarn-site.xml file. Without the manual definition in the configuration file, the MapReduce application master fails to start a YARN container.

Here is the default YARN application CLASSPATH for Linux:

```
$HADOOP_CONF_DIR,
$HADOOP_COMMON_HOME/share/hadoop/common/*,
$HADOOP_COMMON_HOME/share/hadoop/common/lib/*,
$HADOOP_HDFS_HOME/share/hadoop/hdfs/*,
$HADOOP_HDFS_HOME/share/hadoop/hdfs/lib/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/lib/*
```

Here is the default YARN application CLASSPATH for Windows:

```
%HADOOP_CONF_DIR%,
%HADOOP_COMMON_HOME%/share/hadoop/common/*,
%HADOOP_COMMON_HOME%/share/hadoop/common/lib/*,
%HADOOP_HDFS_HOME%/share/hadoop/hdfs/*,
%HADOOP_HDFS_HOME%/share/hadoop/hdfs/lib/*,
%HADOOP_YARN_HOME%/share/hadoop/yarn/*,
%HADOOP_YARN_HOME%/share/hadoop/yarn/lib/*
```

**Note:** On MapR, the YARN application CLASSPATH does not resolve the symbols or variables that are included in pathnames such as SHADOOP_HDFS_HOME.

**Tip:** To apply any change that you make to the cluster, you must restart the node managers.

---

### Change the Trace Level

You can modify the level of tracing by changing the value of the sas.ep.server.trace.level property in the ep-config.xml file. The default value is 4 (TRACE_NOTE).
The `trace-level` represents the level of trace that is produced by the SAS Embedded Process. `trace-level` can be one of the following values:

0
   TRACE_OFF
1
   TRACE_FATAL
2
   TRACE_ERROR
3
   TRACE_WARN
4
   TRACE_NOTE
5
   TRACE_INFO
10
   TRACE_ALL

*Note:* Trace options can produce a significant volume of output. If trace options are not required for troubleshooting or monitoring, set the `trace-level` value to 0.

---

### Adjusting the SAS Embedded Process Performance

You can adjust how the SAS Embedded Process runs by changing or adding its properties. SAS Embedded Process configuration properties can be added to the `mapred-site.xml` configuration on the client side or the `sasep-site.xml` file. If you change the properties in the `ep-config.xml` file located on HDFS, it will affect all SAS Embedded Process jobs. The `ep-config.xml` file is created when you install the SAS Embedded Process.

When using the SAS Embedded Process, there are a number of properties that you can adjust to improve performance. Note that performance properties are valid only when the SAS Embedded Process runs as a MapReduce job.

- You can specify the number of SAS Embedded Process MapReduce tasks per node by changing the `sas.ep.superreader.tasks.per.node` property in the `ep-config.xml` file. The default number of tasks is 6.

The SAS Embedded Process super reader technology does not use the standard MapReduce split calculation. Instead of assigning one split per task, it assigns many. The super reader calculates the splits, groups them, and distributes the groups to a configurable number of mapper tasks based on data locality.
You can specify the number of concurrent nodes that are allowed to run High-Performance Analytics and CAS Parallel Data Connector output tasks by changing the `sas.ep.hpa.output.concurrent.nodes` property.

If this property is set to zero, the SAS Embedded Process allocates tasks on all nodes capable of running a YARN container. If this property is set to –1, the number of concurrent nodes equates to the number of High-Performance Analytics worker nodes. If the number of concurrent nodes exceeds the number of available nodes, the property value is adjusted to the number of available nodes. The default value is 0.

```
<property>
  <name>sas.ep.hpa.output.concurrent.nodes</name>
  <value>number-of-nodes</value>
</property>
```

You can specify the number of High-Performance Analytics and CAS Parallel Data Connector output tasks that are allowed to run per node by changing the `sas.ep.hpa.output.tasks.per.node` property.

The default number of tasks is 1.

```
<property>
  <name>sas.ep.hpa.output.tasks.per.node</name>
  <value>number-of-tasks</value>
</property>
```

You can specify the number of concurrent input reader threads.

Each reader thread takes a file split from the input splits queue, opens the file, positions itself at the beginning of the split, and starts reading the records. Each record is stored on a native buffer that is shared with the DS2 container. When the native buffer is full, it is pushed to the DS2 container for processing. When a reader thread finishes reading a file split, it takes another file split from the input splits queue. The default number of input threads is 3.

```
<property>
  <name>sas.ep.input.threads</name>
  <value>number-of-input-threads</value>
</property>
```

You can specify the number of output writer threads by changing the `sas.ep.output.threads` property.

The SAS Embedded Process super writer technology improves performance by writing output data in parallel, producing multiple parts of the output file per mapper task. Each writer thread is responsible for writing one part of the output file. The default number of output threads is 2.

```
<property>
  <name>sas.ep.output.threads</name>
  <value>number-of-output-threads</value>
</property>
```

You can specify the number of compute threads by changing the `sas.ep.compute.threads` property.

Each compute thread runs one instance of the DS2 program inside the SAS Embedded Process. The DS2 code that runs inside the DS2 container processes the records that it receives. At a given point, DS2 flushes output data to native buffers. The super writer threads take the output data from DS2 buffers and writes them to the super writer thread output file on a designated HDFS location. When all file input
splits are processed and all output data is flushed and written to HDFS, the mapper
task ends. The default number of compute threads is 1.

```xml
<property>
  <name>sas.ep.compute.threads</name>
  <value>number-of-threads</value>
</property>
```

• You can specify the number of buffers that are used for output data by changing the
  sas.ep.output.buffers property in the mapred-site.xml file.

  The number of output buffers should not be less than sas.ep.compute.threads plus
  sas.ep.output.threads. The default number of buffers is 3.

```xml
<property>
  <name>sas.ep.output.buffers</name>
  <value>number-of-buffers</value>
</property>
```

• You can specify the number of native buffers that are used to cache input data by
  changing the sas.ep.input.buffers property in the mapred-site.xml file. The default
  value is 4. The number of input buffers should not be less than
  sas.ep.compute.threads plus sas.ep.input.threads.

```xml
<property>
  <name>sas.ep.input.buffers</name>
  <value>number-of-buffers</value>
</property>
```

• You can specify the optimal size of one input buffer by changing the
  sas.ep.optimal.input.buffer.size property in the mapred-site.xml file.

  The optimal row array size is calculated based on the optimal buffer size. The default
  value is 1 MB.

```xml
<property>
  <name>sas.ep.optimal.input.buffer.size.mb</name>
  <value>number-in-megabytes</value>
</property>
```

---

**Specify the Amount of Memory That the SAS Embedded Process Uses**

The SAS Embedded Process is managed by the Hadoop MapReduce framework. Load
balancing and resource allocation are managed by YARN. Adjust the YARN container
limits to change the amount of memory that the SAS Embedded Process.

For information about how CAS uses memory, see “Memory” in SAS Cloud Analytic
Services: Fundamentals.
Additional Configuration for All Hadoop Distributions – SAS 9.4

Depending on which in-database that product you are running on Base SAS with the SAS Embedded Process, you might get a java.lang.OutOfMemoryError: PermGen space error. This indicates that the Java Virtual Machine PermGen space is exhausted. The PermGen space is not part of the Java Heap memory. It is used to hold metadata that describes the classes and method used by an application.

The PermGen size can be increased to alleviate this issue by adding this line to JREOPTIONS in the SAS configuration file:

```
-jreoptions "(-XX:MaxPermSize=256m)"
```

Additional Configuration for MapR Version 6.0

In Version 6.0 of MapR, the mapreduce.jobhistory properties have a value of 0.0.0.0 for the node where your jobhistory server is running.

To make sure the SAS Embedded Process runs correctly, you must substitute the node where your jobhistory server is running for 0.0.0.0 in all of the mapreduce.jobhistory properties in the mapred-site.xml file. Here are some examples where the node name is maprsrv3.unx.comp.com:

```
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>maprsrv3.unx.comp.com:10020</value>
</property>
<property>
  <name>mapreduce.jobhistory.webapp.address</name>
  <value>maprsrv3.unx.comp.com:19888</value>
</property>
<property>
  <name>mapreduce.jobhistory.webapp.https.address</name>
  <value>maprsrv3.unx.comp.com:19890</value>
</property>
```

Additional Configuration to Access Amazon S3 Object Storage

There are different ways to configure the Hadoop Amazon Web Services (AWS) and S3 object storage. For more information, see Hadoop-AWS module: Integration with Amazon Web Services.

S3A is an Apache Hadoop connector to Amazon S3. S3A supports multiple authentication mechanisms, and can be configured as to which mechanisms to use and the order in which to use them.
There are two ways to configure credentials for the Apache Hadoop S3A client using temporary AWS credentials provider:

- Authenticate by setting the following Hadoop configuration properties in the core-site.xml file.

**Table 6.1  Hadoop Configuration Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Property setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.s3a.access.key</td>
<td>AWS access key ID</td>
<td>&lt;property&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;name&gt;fs.s3a.access.key&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt;your-key&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/property&gt;</td>
</tr>
<tr>
<td>fs.s3a.secret.key</td>
<td>AWS secret key</td>
<td>&lt;property&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;name&gt;fs.s3a.secret.key&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt;your-secret-key&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/property&gt;</td>
</tr>
<tr>
<td>fs.s3a.session.token</td>
<td>the session token, when using org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider as one of the providers</td>
<td>&lt;property&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;name&gt;fs.s3a.session.token&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt;your-token&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/property&gt;</td>
</tr>
<tr>
<td>fs.s3a.aws.credentials.provider</td>
<td>a comma-separated class names of credential provider classes</td>
<td>&lt;property&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;name&gt;fs.s3a.aws.credentials.provider&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt;org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/property&gt;</td>
</tr>
<tr>
<td>fs.s3a.endpoint</td>
<td>AWS region</td>
<td>&lt;property&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;name&gt;fs.s3a.endpoint&lt;/name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt;s3.us-east-1.amazonaws.com&lt;/value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/property&gt;</td>
</tr>
</tbody>
</table>

- Authenticate by using environment variables

S3A supports configuration via the standard AWS environment variables. A full description of AWS environment variables can be found at Configuring the AWS CLI: Environment Variables.

The following environment variables can be used to set the authentication credentials instead of properties in the Hadoop configuration:

**Table 6.2  Hadoop Configuration Properties**

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
<th>Example of Setting the Environment Variable in a SAS Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS_ACCESS_KEY_ID</td>
<td>AWS access key</td>
<td>option set AWS_ACCESS_KEY_ID=&quot;your-key&quot;;</td>
</tr>
</tbody>
</table>
### Environment Variable

**AWS_SECRET_ACCESS_KEY**
- **Description**: AWS secret key. Access and secret key variables override credentials that are stored in credential and configuration files.
- **Example of Setting the Environment Variable in a SAS Session**: `option set=AWS_SECRET_ACCESS_KEY="your-secret-key";`

**AWS_SESSION_TOKEN**
- **Description**: A session token if you are using temporary security credentials.
- **Example of Setting the Environment Variable in a SAS Session**: `option set=AWS_SESSION_TOKEN="your-token";`

**AWS_DEFAULT_REGION**
- **Description**: AWS region. This variable overrides the default region of the in-use profile, if set.
- **Example of Setting the Environment Variable in a SAS Session**: `option set=AWS_DEFAULT_REGION="region-name";`

---

### Additional Configuration When Running Spark on SAS Viya 3.4

To enable the SAS Embedded Process to work on SAS Viya 3.4 when running as a Spark application, you must add the following property to the Spark configuration file, `spark-defaults.conf` that is stored on the CAS controller node.

```
spark.executor.extraClassPath=path-to-sasep_jar_file
```

The `path-to-sasep_jar_file` is the location of the SAS Embedded Process JAR file on the cluster where the SAS Embedded Process is installed. Here is an example.

```
spark.executor.extraClassPath=/opt/sasep/home/jars/sasep.jar
```

To find the location of the SAS Embedded Process JAR file on the cluster, list the SAS Embedded Process configuration file, `ep-config.xml`, that is stored on HDFS in the `/sas/ep/config/` directory.

1. Log to one of the nodes of the cluster.
2. Enter this command to list the SAS Embedded Process configuration file.
   ```
   hadoop fs -cat /sas/ep/config/ep-config.xml
   ```
   The XML file listing will appear.
3. Locate the `sas.ep.home` property which is the SAS Embedded Process home folder. Here is an example.
   ```
   <property>
   <name>sas.ep.home</name>
   <value>/opt/sasep/home</value>
   </property>
   ```
4. Take the property value and append `/jars/sasep.jar`. 
Using the example above, the path would become /opt/sasep/home/jars/sasep.jar.

5. On the CAS controller node, add the following property to the Spark configuration file, spark-defaults.conf.

    spark.executor.extraClassPath=/opt/sasep/SASEPHome/jars/sasep.jar
Part 6

Administrator’s Guide for Netezza

Chapter 7

Netezza Administrator’s Guide ............................... 67
Chapter 7
Netezza Administrator’s Guide

Prerequisites .......................................................... 67
Overview of the In-Database Deployment Package for Netezza ............. 68
Function Publishing Process in Netezza .................................. 68
Netezza Installation and Configuration .................................. 69
  Netezza Installation and Configuration Steps ............................. 69
  Upgrading the SAS Formats Library and Binary Files ....................... 69
  Installing the SAS Formats Library and Binary Files .................... 70
  Running the %INDNZ_PUBLISH_JAZLIB Macro ......................... 70
  Running the %INDNZ_PUBLISH_COMPILEUDF Macro .................. 73
NZCM Commands for the SAS Embedded Process .......................... 76
Netezza Permissions .................................................... 77
Documentation for Using In-Database Processing in Netezza ................ 79

Prerequisites

The in-database deployment package for Netezza requires the following:

• SAS Foundation and the SAS/ACCESS Interface to Netezza. This software must be
  installed before you install and configure the in-database deployment package for
  Netezza.

• a specific version of the Netezza client and server environment. For more
  information, see the SAS Foundation system requirements documentation for your
  operating environment.

• the SAS Embedded Process. The SAS Embedded Process is a SAS server process
  that runs within Netezza to read and write data. The SAS Embedded Process
  contains macros, run-time libraries, and other software that is installed on your
  Netezza system. For more information, see SAS Embedded Process: Deployment
  Guide.
Overview of the In-Database Deployment Package for Netezza

The in-database deployment package for Netezza contains the SAS formats library and two pre-complied binaries for utility functions. The in-database deployment package is available for installation after you deploy SAS/ACCESS Interface to Netezza.

The in-database deployment package for Netezza must be installed and configured before you can perform the following tasks:

- Use the `%INDNZ_PUBLISH_FORMATS` format publishing macro to create or publish the SAS_PUT( ) function and to create or publish user-defined formats as format functions inside the database.
- Use the `%INDNZ_PUBLISH_MODEL` scoring publishing macro to create scoring model functions inside the database.

For more information about using the format and scoring publishing macros, see the [SAS In-Database Products: User’s Guide](#).

The SAS formats library is a run-time library that is installed on your Netezza server. This installation is done so that the SAS scoring model functions and the SAS_PUT( ) function can access the routines within the run-time library. The SAS formats library contains the formats that are supplied by SAS.

The `%INDNZ_PUBLISH_JAZLIB` macro registers the SAS formats library. The `%INDNZ_PUBLISH_COMPILEUDF` macro registers a utility function in the database. The utility function is then called by the format and scoring publishing macros. You must run these two macros before you run the format and scoring publishing macros.

Function Publishing Process in Netezza

To publish the SAS scoring model functions, the SAS_PUT( ) function, and format functions on Netezza systems, the format and scoring publishing macros perform the following tasks:

- Create and transfer the files, using the Netezza External Table interface, to the Netezza server.
  
  Using the Netezza External Table interface, the source files are loaded from the client to a database table through remote ODBC. The source files are then exported to files (external table objects) on the host. Before transfer, each source file is divided into 32K blocks and converted to hexadecimal values to avoid problems with special characters, such as line feed or quotation marks. After the files are exported to the host, the source files are converted back to text.

- Compile those source files into object files using a Netezza compiler.
- Link with the SAS formats library.
- Register those object files with the Netezza system.

*Note:* This process is valid only when using publishing formats and scoring functions. It is not applicable to the SAS Embedded Process. If you use the SAS Embedded
Process, the scoring publishing macro creates the scoring files and uses the SAS/ACCESS Interface to Netezza to insert the scoring files into a model table.

**Netezza Installation and Configuration**

**Netezza Installation and Configuration Steps**

1. If you are upgrading from a previous version, follow the instructions in “Upgrading the SAS Formats Library and Binary Files” on page 69.

2. Install the SAS formats library and the binary files.
   
   For more information, see “Installing the SAS Formats Library and Binary Files” on page 70.

3. Run the `%INDNZ_PUBLISH_JAZLIB` macro to publish the SAS formats library as an object.
   
   For more information, see “Running the `%INDNZ_PUBLISH_JAZLIB` Macro” on page 70.

4. Run the `%INDNZ_PUBLISH_COMPILEUDF` macro.
   
   For more information, see “Running the `%INDNZ_PUBLISH_COMPILEUDF` Macro” on page 73.

5. If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, perform the additional configuration tasks in Chapter 12, “Configuring SAS Model Manager,” on page 115.

**Upgrading the SAS Formats Library and Binary Files**

To upgrade from a previous version of the SAS Formats Library and binary files, follow these steps.

1. Run the `%INDNZ_PUBLISH_JAZLIB` macro with ACTION=DROP to remove the SAS formats library as an object.
   
   For more information, see “Running the `%INDNZ_PUBLISH_JAZLIB` Macro” on page 70.

2. Run the `%INDNZ_PUBLISH_COMPILEUDF` macro with ACTION=DROP to remove the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions.
   
   For more information, see “Running the `%INDNZ_PUBLISH_COMPILEUDF` Macro” on page 73.

3. Navigate to the `/nz/extensions/SAS` directory and delete the `SASFormatsLibraryForNetezza` directory.

4. Continue the installation instructions in “Installing the SAS Formats Library and Binary Files” on page 70.
Installing the SAS Formats Library and Binary Files

The SAS formats library and the binary files for the SAS_COMPILEUDF function are contained in a self-extracting archive file. The self-extracting archive file is located in the `YourSASHome/SASFormatsLibraryForNetezza/3.1/Netezza32bitTwinFin/` directory.

To move and unpack the self-extracting archive file, follow these steps:

1. Using a method of your choice, transfer the `accelnetzfmt-3.1-n_lax.sh` to a location on your Netezza server.
   
   *`n` is a number that indicates the latest version of the file. If this is the initial installation, `n` has a value of 1. Each time you re-install or upgrade, `n` is incremented by 1.*

2. After the `accelnetzfmt-3.1-n_lax.sh` file has been transferred to the Netezza server, log on as the user who owns the Netezza appliance (usually the “nz” ID).

3. Use the following commands at the UNIX prompt to unpack the self-extracting archive file.

   ```
   mkdir -p /nz/extensions
   chmod 755 /nz/extensions
   cd /nz/extensions
   chmod 755 path_to_sh_file/accelnetzfmt-3.1-n_lax.sh
   path_to_sh_file is the location to which you copied the self-extracting archive file in Step 1.
   ```

   After the script runs and the files are unpacked, the target directories should look similar to these.

   ```
   /nz/extensions/SAS/SASFormatsLibraryForNetezza/3.1-n/bin/InstallAccelNetzFmt.sh
   /nz/extensions/SAS/SASFormatsLibraryForNetezza/3.1-n/lib/SAS_CompileUDF.o_spu10
   /nz/extensions/SAS/SASFormatsLibraryForNetezza/3.1-n/lib/SAS_CompileUDF.o_x86
   /nz/extensions/SAS/SASFormatsLibraryForNetezza/3.1-n/lib/libjazxfbrs_spu10.so
   /nz/extensions/SAS/SASFormatsLibraryForNetezza/3.1-n/lib/libjazxfbrs_x86.so
   ```

   There also is a symbolic link such that `/nz/extensions/SAS/SASFormatsLibraryForNetezza/3.1` points to the latest version.

Running the `%INDNZ_PUBLISH_JAZLIB` Macro

Overview of Publishing the SAS Formats Library

The SAS formats library is a shared library and must be published and registered as an object in the Netezza database. The library is linked to the scoring and format publishing macros through a DEPENDENCIES statement when the scoring model functions or formats are created.

You must run the `%INDNZ_PUBLISH_JAZLIB` macro to publish and register the SAS formats library. The `%INDNZ_PUBLISH_JAZLIB` macro publishes and registers the SAS formats library in the database as the `sas_jazlib` object.

%INDNZ_PUBLISH_JAZLIB Macro Run Process

To run the `%INDNZ_PUBLISH_JAZLIB` macro, follow these steps:
1. Start SAS and submit the following command in the Enhanced Editor or Program Editor:

```
%let indconn=SERVER=yourservername USER=youruserid PW=yourpwd DB=database;
```

For more information, see the “INDCONN Macro Variable” on page 71.

2. Run the %INDNZ_PUBLISH_JAZLIB macro. For more information, see “%INDNZ_PUBLISH_JAZLIB Macro Syntax” on page 72.

**INDCONN Macro Variable**

The INDCONN macro variable is used to provide credentials to connect to Netezza. You must specify server, user, password, and database information to access the machine on which you have installed the Netezza data warehouse. You must assign the INDCONN macro variable before the %INDNZ_PUBLISH_JAZLIB macro is invoked.

The value of the INDCONN macro variable for the %INDNZ_PUBLISH_JAZLIB macro has this format:

```
SERVER=server USER=userid PASSWORD=password DATABASE=database SCHEMA=schema-name
```

**SERVER=server | 'server'**

specifies the server name or IP address of the server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, enclose the server name in quotation marks.

**USER=userid | 'userid'**

specifies the Netezza user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, enclose the user name in quotation marks.

**PASSWORD=password | 'password'**

specifies the password that is associated with your Netezza user name. If the password contains spaces or nonalphanumeric characters, enclose the password in quotation marks.

Tip Use only PASSWORD=, PASS=, or PW= for the password argument. PWD= is not supported and causes an error.

**DATABASE=database | 'database'**

specifies the name of the database on the server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, enclose the database name in quotation marks.

**SCHEMA=schema-name | 'schema-name'**

specifies the name of the schema where the SAS formats library is published.

**Restriction**

This argument is supported only on Netezza 7.0.3 or later.
Interaction

The schema that is specified by the `%INDNZ_PUBLISH_JAZLIB` macro’s DBSCHEMA= argument takes precedence over the schema that you specify in the INDCONN macro variable. If you do not specify a schema in the DBSCHEMA= argument or the INDCONN macro variable, the default schema for the target database is used.

%INDNZ_PUBLISH_JAZLIB Macro Syntax

```sas
%INDNZ_PUBLISH_JAZLIB
  (<DATABASE=database>
   <, DBSCHEMA=schema-name>
   <, ACTION=CREATE | REPLACE | DROP>
   <, OUTDIR=diagnostic-output-directory>
  );
```

Arguments

**DATABASE=**

*database*
specifies the name of a Netezza database to which the SAS formats library is published as the `sas_jazlib` object.

- **Default**: SASLIB
- **Interaction**: The database that is specified by the DATABASE= argument takes precedence over the database that you specify in the INDCONN macro variable.
- **Tip**: The object name for the SAS formats library is `sas_jazlib`.

**DBSCHEMA=**

*schema-name*
specifies the name of a Netezza schema to which the SAS formats library is published.

- **Restrictions**: This argument is supported only on Netezza v7.0.3 or later.
- **Interaction**: The schema that is specified by the DBSCHEMA= argument takes precedence over the schema that you specify in the INDCONN macro variable. If you do not specify a schema in the DBSCHEMA= argument or the INDCONN macro variable, the default schema for the target database is used.

**ACTION=**

*CREATE | REPLACE | DROP*
specifies that the macro performs one of the following actions:

- **CREATE**: creates a new SAS formats library.
- **REPLACE**: overwrites the current SAS formats library, if a SAS formats library by the same name is already registered, or creates a new SAS formats library if one is not registered.
- **DROP**: causes the SAS formats library to be dropped from the Netezza database.

- **Default**: CREATE
If the SAS formats library was published previously and you specify ACTION=CREATE, you receive warning messages that the library already exists. You are prompted to use REPLACE. If you specify ACTION=DROP and the SAS formats library does not exist, you receive an error message.

OUTDIR=diagnostic-output-directory
specifies a directory that contains diagnostic files.

Files that are produced include an event log that contains detailed information about the success or failure of the publishing process.

Running the %INDNZ_PUBLISH_COMPILEUDF Macro

Overview of the %INDNZ_PUBLISH_COMPILEUDF Macro

The %INDNZ_PUBLISH_COMPILEUDF macro creates three functions:

• SAS_COMPILEUDF. This function facilitates the scoring and format publishing macros. The SAS_COMPILEUDF function compiles the scoring model and format source files into object files. This compilation uses a Netezza compiler and occurs through the SQL interface.

• SAS_DIRECTORYUDF and SAS_HEXTOTEXTUDF. These functions are used when the scoring and format publishing macros transfer source files from the client to the host using the Netezza External Tables interface. SAS_DIRECTORYUDF creates and deletes temporary directories on the host. SAS_HEXTOTEXTUDF converts the files from hexadecimal back to text after the files are exported on the host. For more information about the file transfer process, see “Function Publishing Process in Netezza” on page 68.

You have to run the %INDNZ_PUBLISH_COMPILEUDF macro only one time. The SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions must be published before the %INDNZ_PUBLISH_FORMATS or %INDNZ_PUBLISH_MODEL macros are run. Otherwise, these macros fail.

Note: To publish the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions, you must have the appropriate Netezza user permissions to create these functions in either the SASLIB database (default) or in the database that is used in lieu of SASLIB. For more information, see “Netezza Permissions” on page 77.

%INDNZ_PUBLISH_COMPILEUDF Macro Run Process

To run the %INDNZ_PUBLISH_COMPILEUDF macro to publish the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions, follow these steps:

1. Create either a SASLIB database or a database to be used in lieu of the SASLIB database.

   This database is where the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions are published. You specify this database in the DATABASE argument of the %INDNZ_PUBLISH_COMPILEUDF macro. For more information about how to specify the database that is used in lieu of SASLIB, see “%INDNZ_PUBLISH_COMPILEUDF Macro Syntax” on page 75.
2. Start SAS and submit the following command in the Enhanced Editor or Program Editor.

```sas
%let indconn = server=yourserver user=youruserid password=yourpwd
database=database;
```

For more information, see the “INDCONN Macro Variable” on page 74.

3. Run the %INDNZ_PUBLISH_COMPILEUDF macro. For more information, see “%INDNZ_PUBLISH_COMPILEUDF Macro Syntax” on page 75.

After the SAS_COMPILEUDF function is published, the model or format publishing macros can be run to publish the scoring model or format functions.

**INDCONN Macro Variable**

The INDCONN macro variable provides the credentials to make a connection to Netezza. You must specify the server, user, password, and database information to access the machine on which you have installed the Netezza database. You must assign the INDCONN macro variable before the %INDNZ_PUBLISH_COMPILEUDF macro is invoked.

The value of the INDCONN macro variable for the %INDNZ_PUBLISH_COMPILEUDF macro has this format.

```
SERVER=server USER=userid PASSWORD=password DATABASE=SASLIB | database SCHEMA=schema-name
```

- **SERVER=server | 'server'**
  Specifies the server name or IP address of the server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, enclose the server name in quotation marks.

- **USER=userid | 'userid'**
  Specifies the Netezza user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, enclose the user name in quotation marks.

- **PASSWORD=password | 'password'**
  Specifies the password that is associated with your Netezza user name. If the password contains spaces or nonalphanumeric characters, enclose the password in quotation marks.

  Tip Use only PASSWORD=, PASS=, or PW= for the password argument. PWD= is not supported and causes an error.

- **DATABASE=SASLIB | database | 'database'**
  Specifies the name of the database on the server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, enclose the database name in quotation marks.

  Default SASLIB

  Interactions The database that is specified by the %INDNZ_PUBLISH_COMPILEUDF macro’s DATABASE= argument takes precedence over the database that you specify in the INDCONN macro variable. If you do not specify a value for DATABASE= in either the INDCONN macro variable or the %INDNZ_PUBLISH_COMPILEUDF macro, the default value of
SASLIB is used. For more information, see
“%INDNZ_PUBLISH_COMPILEUDF Macro Syntax” on page 75.

If the SAS_COMPILEUDF function is published in a database other than SASLIB, then that database name should be used instead of SASLIB for the DBCOMPILE argument in the
%INDNZ_PUBLISH_FORMATS and %INDNZ_PUBLISH_MODEL macros. Otherwise, the %INDNZ_PUBLISH_FORMATS and %INDNZ_PUBLISH_MODEL macros fail when calling the SAS_COMPILEUDF function during the publishing process. If a database name is not specified, the default is SASLIB. For documentation on the %INDNZ_PUBLISH_FORMATS and %INDNZ_PUBLISH_MODEL macros, see “Documentation for Using In-Database Processing in Netezza” on page 79.

SCHEMA=schema-name | 'schema-name'
specifies the name of the schema where the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions are published.

Restriction This argument is supported only on Netezza v7.0.3 or later.

Interaction The schema that is specified by the
%INDNZ_PUBLISH_COMPILEUDF macro’s DBSCHEMA= argument takes precedence over the schema that you specify in the INDCONN macro variable. If you do not specify a schema in the DBSCHEMA= argument or the INDCONN macro variable, the default schema for the target database is used.

%INDNZ_PUBLISH_COMPILEUDF Macro Syntax
%INDNZ_PUBLISH_COMPILEUDF
(<DATABASE=database-name>
<, DBSCHEMA=schema-name>
<, ACTION=CREATE | REPLACE | DROP>
<, OUTDIR=diagnostic-output-directory>
);

Arguments
DATABASE=database-name
specifies the name of a Netezza database to which the SAS_COMPILEUDF is published.

Default SASLIB

Interaction The database that is specified by the DATABASE= argument takes precedence over the database that you specify in the INDCONN macro variable. For more information, see “INDCONN Macro Variable” on page 74.

DBSCHEMA=schema-name
specifies the name of a Netezza schema to which the SAS_COMPILEUDF function is published.

Restriction This argument is supported only on Netezza v7.0.3 or later.

Interaction The schema that is specified by the DBSCHEMA= argument takes precedence over the schema that you specify in the INDCONN macro
variable. If you do not specify a schema in the DBSCHEMA= argument or the INDCONN macro variable, the default schema for the target database is used.

**ACTION=CREATE | REPLACE | DROP**
specifies that the macro performs one of the following actions:

**CREATE**
creates a new SAS_COMPILEUDF function.

**REPLACE**
overwrites the current SAS_COMPILEUDF function, if a SAS_COMPILEUDF function by the same name is already registered, or creates a new SAS_COMPILEUDF function if one is not registered.

**DROP**
causes the SAS_COMPILEUDF function to be dropped from the Netezza database.

Default  CREATE

**Tip**  If the SAS_COMPILEUDF function was published previously and you specify ACTION=CREATE, you receive warning messages that the function already exists and be prompted to use REPLACE. If you specify ACTION=DROP and the SAS_COMPILEUDF function does not exist, you receive an error message.

**OUTDIR=** diagnostic-output-directory
specifies a directory that contains diagnostic files.

**Tip**  Files that are produced include an event log that contains detailed information about the success or failure of the publishing process.

---

## NZCM Commands for the SAS Embedded Process

The following table lists and describes the NZCM commands that you can use with the SAS Embedded Process.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm -help</td>
<td>Displays help for NZCM commands</td>
</tr>
<tr>
<td>nzcm --installed</td>
<td>Displays the filename (SASTKInDatabaseServerForNetezza) and the version number that is installed</td>
</tr>
<tr>
<td>nzcm --registered nzcm -r</td>
<td>Displays the filename (SASTKInDatabaseServerForNetezza) and the version number that is registered</td>
</tr>
<tr>
<td>nzcm --unregister SASTKInDatabaseServerForNetezza nzcm -u SASTKInDatabaseServerForNetezza</td>
<td>Unregisters the SAS Embedded Process</td>
</tr>
<tr>
<td>Command</td>
<td>Action performed</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>nzcm --unregister sas_ep</td>
<td>Unregisters the sas_ep cartridge</td>
</tr>
<tr>
<td>nzcm -u sas_ep</td>
<td>Unregisters the sas_ep cartridge</td>
</tr>
</tbody>
</table>

Note: The sas_ep cartridge is installed only once. It does not need to be unregistered or uninstalled when the SAS Embedded Process is upgraded or re-installed. The sas_ep cartridge needs to be unregistered and uninstalled only when Netezza changes the cartridge version.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm -uninstall SASTKInDatabaseServerForNetezza</td>
<td>Uninstalls the SAS Embedded Process</td>
</tr>
<tr>
<td>nzcm -e SASTKInDatabaseServerForNetezza</td>
<td>Uninstalls the SAS Embedded Process</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm --uninstall sas_ep</td>
<td>Uninstalls the sas_ep cartridge</td>
</tr>
<tr>
<td>nzcm -e sas_ep</td>
<td>Uninstalls the sas_ep cartridge</td>
</tr>
</tbody>
</table>

Note: The sas_ep cartridge is installed only once. It does not need to be unregistered or uninstalled when the SAS Embedded Process is upgraded or re-installed. The sas_ep cartridge needs to be unregistered and uninstalled only when Netezza changes the cartridge version.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm --install SSASTKInDatabaseServerForNetezza-13.0.0.0.nzc</td>
<td>Installs the SAS Embedded Process</td>
</tr>
<tr>
<td>nzcm -i SSASTKInDatabaseServerForNetezza-13.0.0.0.nzc</td>
<td>Installs the SAS Embedded Process</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm --install sas_ep</td>
<td>Installs the sas_ep cartridge</td>
</tr>
<tr>
<td>nzcm -i sas_ep</td>
<td>Installs the sas_ep cartridge</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm --register SASTKInDatabaseServerForNetezza</td>
<td>Registers the SAS Embedded Process</td>
</tr>
<tr>
<td>nzcm -r SASTKInDatabaseServerForNetezza</td>
<td>Registers the SAS Embedded Process</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>nzcm --register sas_ep</td>
<td>Registers the sas_ep cartridge</td>
</tr>
<tr>
<td>nzcm -register sas_ep</td>
<td>Registers the sas_ep cartridge</td>
</tr>
</tbody>
</table>

---

**Netezza Permissions**

There are three sets of permissions involved with the in-database software.

- The first set of permissions is needed by the person who publishes the SAS formats library and the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and SAS_HEXTOTEXTUDF functions. These permissions must be granted before the %INDNZ_PUBLISH_JAZLIB and %INDNZ_PUBLISH_COMPILEUDF macros are run. Without these permissions, running these macros fails.

The following table summarizes the permissions that are needed by the person who publishes the formats library and the functions.
Table 7.1 Permissions Needed to Publish the SAS Formats Library and Functions

<table>
<thead>
<tr>
<th>Permission Needed</th>
<th>Authority Required to Grant Permission</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE LIBRARY permission to run the %INDNZ_PUBLISH_JAZLIB macro that publishes the SAS formats library (sas_jazlib object)</td>
<td>System Administrator or Database Administrator</td>
<td>GRANT CREATE LIBRARY TO fmtlib-publisher-id</td>
</tr>
<tr>
<td>CREATE FUNCTION permission to run the %INDNZ_PUBLISH_COMPILEUDF macro that publishes the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and the SAS_HEXTOTEXTUDF functions</td>
<td>Note: If you have SYSADM or DBADM authority, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions.</td>
<td>GRANT CREATE FUNCTION TO compileudf-publisher-id</td>
</tr>
</tbody>
</table>

- The second set of permissions is needed by the person who runs the format publishing macro, %INDNZ_PUBLISH_FORMATS, or the scoring publishing macro, %INDNZ_PUBLISH_MODEL. The person who runs these macros is not necessarily the same person who runs the %INDNZ_PUBLISH_JAZLIB and %INDNZ_PUBLISH_COMPILEUDF macros. These permissions are most likely needed by the format publishing or scoring model developer. Without these permissions, the publishing of the scoring model functions and the SAS_PUT( ) function and formats fails.

  Note: Permissions must be granted for every format and scoring model publisher and for each database that the format and scoring model publishing uses. Therefore, you might need to grant these permissions multiple times. After the Netezza permissions are set appropriately, the format and scoring publishing macros can be run.

  Note: When permissions are granted to specific functions, the correct signature, including the sizes for numeric and string data types, must be specified.

The following table summarizes the permissions that are needed by the person who runs the format or scoring publishing macro.
Table 7.2  Permissions Needed to Run the Format or Scoring Publishing Macro

<table>
<thead>
<tr>
<th>Permission Needed</th>
<th>Authority Required to Grant Permission</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE permission for the SAS Formats Library</td>
<td>System Administrator or Database Administrator</td>
<td>GRANT EXECUTE ON SAS_JAZLIB TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>EXECUTE permission for the SAS_COMPILEUDF function</td>
<td>Note: If you have SYSADM or DBADM authority, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions.</td>
<td>GRANT EXECUTE ON SAS_COMPILEUDF TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>EXECUTE permission for the SASDIRECTORYUDF function</td>
<td></td>
<td>GRANT EXECUTE ON SAS_DIRECTORYUDF TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>EXECUTE permission for the SAS_HEXTOTEXTUDF function</td>
<td></td>
<td>GRANT EXECUTE ON SAS_HEXTOTEXTUDF TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td>CREATE FUNCTION, CREATE TABLE, CREATE TEMP TABLE, and CREATE EXTERNAL TABLE permissions to run the format and scoring publishing macros</td>
<td></td>
<td>GRANT CREATE FUNCTION TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRANT CREATE TABLE TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRANT CREATE TEMP TABLE TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRANT CREATE EXTERNAL TABLE TO scoring-or-fmt-publisher-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRANT UNFENCED TO scoring-or-fmt-publisher-id</td>
</tr>
</tbody>
</table>

- The third set of permissions is needed by the person who runs the SAS Embedded Process to create scoring files.

The SAS Embedded Process has a dependency on the IBM Netezza Analytics (INZA) utility. You must grant the user and database permissions using these commands.

```
/nz/export/ae/utlities/bin/create_inza_db_user.sh user-name database-name
/nz/export/ae/utlities/bin/create_inza_db.sh database-name
```

Note: If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, additional permissions are required. For more information, see Chapter 12, “Configuring SAS Model Manager,” on page 115.

Documentation for Using In-Database Processing in Netezza

For information about how to publish SAS formats, the SAS_PUT( ) function, and scoring models, see the SAS In-Database Products: User’s Guide.
Prerequisites

In-database processing for Oracle requires the following:

- SAS Foundation and the SAS/ACCESS Interface to Oracle. This software must be installed before you configure in-database processing for Oracle.

- a specific version of the Oracle client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

- the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Oracle to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your Oracle system. The software is installed so that the SAS scoring files created in Oracle can access the routines within the SAS Embedded Process’s run-time libraries. For more information, see *SAS Embedded Process: Deployment Guide*.

Overview of the SAS Embedded Process for Oracle

The SAS Embedded Process for Oracle must be installed and configured before you perform the following tasks:

- Use the %INDOR_PUBLISH_MODEL scoring publishing macro to create scoring files inside the database.
Run SAS High-Performance Analytics when the analytics cluster is using a parallel connection with a remote Oracle Exadata appliance. The SAS Embedded Process, which resides on the data appliance, is used to provide high-speed parallel data transfer between the data appliance and the analytics environment where it is processed.

For more information, see *SAS High-Performance Analytics: Installation and Configuration Guide*.

For more information about using the scoring publishing macros, see *SAS In-Database Products: User's Guide*.

### Oracle Configuration

To configure Oracle for in-database processing, follow these steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Where to Go for Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create users and objects for the SAS Embedded Process</td>
<td>“Creating Users and Objects for the SAS Embedded Process” on page 84</td>
</tr>
<tr>
<td>2</td>
<td>Review the permissions needed for in-database processing</td>
<td>“Oracle Permissions” on page 85</td>
</tr>
</tbody>
</table>

### Creating Users and Objects for the SAS Embedded Process

After the SAS Embedded Process for Oracle is installed, the DBA must create the users and grant user privileges. The DBA needs to perform these tasks before the SAS administrator can create the objects for the Oracle server. The users and objects are required for the SAS Embedded Process to work.

*Note: SQLPLUS or an equivalent SQL tool can be used to submit the SQL statements in this topic.*

1. Create a SASADMIN user.

   To create the user accounts for Oracle, the DBA must perform the following steps:
   a. Change the directory to `/opt/sas/ep/home/admin`.
   b. Connect as SYS, using the following command:

      ```
      sqlplus sys/<password> as sysdba
      ```
   c. Create and grant user privileges for the SASADMIN user.

   Here is an example of how to create a SASADMIN user.

   ```
   CREATE USER SASADMIN IDENTIFIED BY <password>
   DEFAULT TABLESPACE <tablespace-name>
   TEMPORARY TABLESPACE <tablespace-name>;
   GRANT UNLIMITED TABLESPACE TO SASADMIN;
   ```
d. Submit the following SQL script to grant the required privileges to the SASADMIN user.

    SQL>@sasadmin_grant_privs.sql

e. If the script does not automatically log you off after it is complete, log off from the SQLPLUS session using “Quit” or close your SQL tool.

2. Create the necessary database objects.

   To create the objects and the SASEPFUNC table function that are needed to run the scoring model, the SAS administrator (SASADMIN) must perform the following steps:

   a. Change the current directory to /opt/sas/ep/home/admin (if you are not already there).

   b. Connect as SASADMIN, using the following command:

      sqlplus sasadmin/<password>

   c. Submit the following SQL statement:

      @create_sasepfunc.sql;

      Note: You can ignore the following errors:

      ORA-00942: table or view does not exist
      ORA-01432: public synonym to be dropped does not exist

---

Oracle Permissions

The person who runs the %INDOR_CREATE_MODELTABLE needs CREATE permission to create the model table. Here is an example.

```
GRANT CREATE TABLE TO userid
```

The person who runs the %INDOR_PUBLISH_MODEL macro needs INSERT permission to load data into the model table. This permission must be granted after the model table is created. Here is an example.

```
GRANT INSERT ON model-table-name TO user-id
```

Note: The RESOURCE user privilege that was granted in the previous topic includes the permissions for CREATE, DELETE, DROP, and INSERT.

If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, additional permissions are required. For more information, see Chapter 12, “Configuring SAS Model Manager,” on page 115.

---

Documentation for Using In-Database Processing in Oracle

For information about how to publish SAS scoring models, see the \textit{SAS In-Database Products: User’s Guide}. 
Part 8

Administrator’s Guide for SAP HANA

Chapter 9
SAP HANA Administrator’s Guide
Chapter 9

SAP HANA Administrator’s Guide

Prerequisites

In-database processing for SAP HANA requires the following:

- SAS Foundation and the SAS/ACCESS Interface to SAP HANA. This software must be installed before you configure in-database processing for SAP HANA.

- a specific version of the SAP HANA client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

- the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within SAP HANA to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your SAP HANA system. These installations are done so that the SAS scoring files created in SAP HANA can access routines within the SAS Embedded Process run-time libraries. For more information, see *SAS Embedded Process: Deployment Guide*.

Overview of the SAS Embedded Process for SAP HANA

The SAS Embedded Process for SAP HANA must be installed and configured before you can perform the following tasks:
- Use the `%INDHN_PUBLISH_MODEL` scoring publishing macro to insert a model into the model table in the SAP HANA database.

For more information about using the scoring publishing macros, see the *SAS In-Database Products: User’s Guide*.

- Run SAS High-Performance Analytics when the analytics cluster is using a parallel connection with a remote SAP HANA appliance. The SAS Embedded Process, which resides on the data appliance, is used to provide high-speed parallel data transfer between the data appliance and the analytics environment where it is processed.

For more information, see the *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.

---

### SAP HANA Configuration

To configure SAP HANA for in-database processing, follow these steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Where to Go for Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Review controlling the SAS Embedded Process</td>
<td>“Controlling the SAS Embedded Process” on page 90</td>
</tr>
<tr>
<td>2</td>
<td>Review the semaphore requirements</td>
<td>“Semaphore Requirements When Using the SAS Embedded Process for SAP HANA” on page 91</td>
</tr>
<tr>
<td>3</td>
<td>Review the permissions needed for in-database processing</td>
<td>“SAP HANA Permissions” on page 91</td>
</tr>
<tr>
<td>4</td>
<td>If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, perform additional configuration tasks</td>
<td>Chapter 12, “Configuring SAS Model Manager,” on page 115.</td>
</tr>
</tbody>
</table>

---

### Controlling the SAS Embedded Process

The SAS Embedded Process starts when you run the `StartupSASEP.sh` script. It continues to run until it is manually stopped or the database is shut down.

*Note:* Starting and stopping the SAS Embedded Process has implications for all scoring model publishers.

*Note:* Manually starting and stopping the SAS Embedded Process requires HANA database administrator user permissions.

When the SAS Embedded Process is installed, the `ShutdownSASEP.sh` and `StartupSASEP.sh` scripts are installed in the `/opt/sas/ep/home/bin` directory. For more information about these files, see “Deploying SAS Embedded Process for SAP HANA” in *SAS Embedded Process: Deployment Guide*.

Use the following command to start the SAS Embedded Process:
Note: The `-verbose` option is on by default and provides a status of the start-up operations as they occur. You can specify the `-quiet` option to suppress messages.

ShutdownSASEP.sh shuts down the SAS Embedded Process. It is designed to be used to shut down the SAS Embedded Process prior to a database upgrade or re-install. This script should not be used as part of the normal operation.

Use the following command to shut down the SAS Embedded Process.

`. /ShutdownSASEP.sh`

Note: The `-verbose` option is on by default and provides a status of the shutdown operations as they occur. You can specify the `-quiet` option to suppress messages.

---

**Semaphore Requirements When Using the SAS Embedded Process for SAP HANA**

Each time a query using the SAS_EP stored procedure is invoked to execute a score, it requests a set of semaphore arrays (sometimes referred to as semaphore "sets") from the operating system. The SAS Embedded Process releases the semaphore arrays back to the operating system after scoring is complete.

The SAP HANA server that runs the SAS Embedded Process should be configured with a minimum of 1024 to 2048 semaphore arrays.

Note: The semaphore limit on the “maximum number of arrays” is distinct from the semaphore limit on the “maximum number of semaphores system wide”. The Linux `ipcs -sl` command shows the typical default semaphore-related limits set on SAP HANA:

```
------- Semaphore Limits -------
max number of arrays = 2048
max semaphores per array = 250
max semaphores system wide = 512000
max ops per semop call = 100
semaphore max value = 32767
```

---

**SAP HANA Permissions**

The following permissions are needed by the person who installs the in-database deployment package:

Note: Some of the permissions listed below cannot be granted until the Auxiliary Wrapper Generator and Eraser Procedures are installed. For more information, see “SASLINK AFL Plug-ins and Auxiliary Wrapper Procedures” in SAS Embedded Process: Deployment Guide.
Table 9.1  Permissions Needed to Install the SAP HANA In-Database Deployment Package

<table>
<thead>
<tr>
<th>Task</th>
<th>Permission Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack the self-extracting archive file</td>
<td>owner of the SAS Embedded Process install directory. The SAS Embedded Process install directory must have permissions that allow Read and Execute permission by the database administrator user.</td>
</tr>
<tr>
<td>Install or uninstall the SAS Embedded Process (run InstallSASEPFiles.sh or UninstallSASEPFiles.sh script)</td>
<td>root authority</td>
</tr>
<tr>
<td>Import the SAS_EP procedure package</td>
<td>a user on the SAP HANA server that has at least the CONTENT_ADMIN role or its equivalent</td>
</tr>
<tr>
<td>Install AFL plugins (requires starting and stopping the database)</td>
<td>root authority and database administrator</td>
</tr>
</tbody>
</table>

The following permissions are needed by the person who runs the scoring models. Without these permissions, the publishing of the scoring models fails:

SAP HANA SPS11:

- AFLPM_CREATOR_ERASER_EXECUTE to userid | role;
- EXECUTE, SELECT, INSERT, UPDATE, and DELETE on the schema that is used for scoring

In addition, the roles of sas.ep::User and AFL_SYS_AFL_SASLINK_AREA_EXECUTE must be assigned to any user who wants to perform in-database processing. The sas.ep::User role is created when you import the SAS_EP stored procedure. The AFL_SYS_AFL_SASLINK_AREA_EXECUTE role is created when the AFL wrapper generator is created.

Note: If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, additional permissions are required. For more information, see Chapter 12, “Configuring SAS Model Manager,” on page 115.

Documentation for Using In-Database Processing in SAP HANA

For information about using in-database processing in SAP HANA, see the SAS In-Database Products: User’s Guide.
Part 9

Administrator’s Guide for Teradata

Chapter 10
  Teradata Administrator’s Guide ........................................ 95

Chapter 11
  SAS Data Quality Accelerator for Teradata ...................... 103
Chapter 10

Teradata Administrator’s Guide

Prerequisites

The SAS Embedded Process deployment package for Teradata requires the following:

- SAS Foundation and the SAS/ACCESS Interface to Teradata. This software must be installed before you install and configure the in-database deployment package for Teradata.

- The SAS in-database and high-performance analytic products require a specific version of the Teradata client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

- the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Teradata to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your Teradata system. For more information, see SAS Embedded Process: Deployment Guide.
Overview of the In-Database Deployment Package for Teradata

The in-database deployment package for Teradata contains the SAS formats library and binary files. The in-database deployment package is available after you install SAS/ACCESS Interface to Teradata.

The in-database deployment package for Teradata must be installed and configured before you can perform the following tasks:

- Use the %INDTD_PUBLISH_FORMATS format publishing macro to publish the SAS_PUT( ) function and to publish user-defined formats as format functions inside the database.
  For more information about using the format publishing macros, see the SAS In-Database Products: User’s Guide.

- Use the %INDTD_PUBLISH_MODEL scoring publishing macro to publish scoring model files or functions inside the database.
  For more information about using the scoring publishing macros, see the SAS In-Database Products: User’s Guide.

- Use the SAS In-Database Code Accelerator for Teradata to execute DS2 thread programs in parallel inside the database.
  For more information, see the SAS In-Database Products: User’s Guide.

- Run SAS High-Performance Analytics when the analytics cluster is using a parallel connection with a remote Teradata data appliance. The SAS Embedded Process, which resides on the data appliance, is used to provide high-speed parallel data transfer between the data appliance and the analytics environment where it is processed.
  For more information, see the SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide.

The SAS formats library is a run-time library that you install on your Teradata system. This installation is done so that the SAS scoring model functions or the SAS_PUT( ) function can access the routines within the run-time library. The SAS formats library contains the formats that are supplied by SAS.

Note: The SAS formats library is not required by the SAS Data Quality Accelerator for Teradata.

Note: If you are performing a system expansion where additional nodes are being added, the version of the SAS formats library on the new database nodes must be the same as the version that is being used on already existing nodes.

Teradata Installation and Configuration

Teradata Installation and Configuration Steps

To install the in-database deployment package, follow these steps:
1. If you are upgrading from a previous version, follow the instructions in “Upgrading from a Previous Version” on page 97.

2. Install the in-database deployment package.
   
   For more information, see “Installing the SAS Formats Library and Binary Files” on page 98.

3. If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, perform the additional configuration tasks provided in Chapter 12, “Configuring SAS Model Manager,” on page 115.

Note: If you are using any of the following SAS software, additional configuration is needed:

- SAS High-Performance Analytics
  
  For more information, see SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide.

- SAS Data Quality Accelerator for Teradata
  
  For more information, see Chapter 11, “SAS Data Quality Accelerator for Teradata,” on page 103.

**Upgrading from a Previous Version**

To upgrade from a previous version of the SAS Formats Library follow these steps:

1. Check the current installed version of the SAS formats library.
   
   How you do this depends on the version of the SAS formats library.
   
   - If a SAS 9.2 version of the formats library is currently installed, run this command:
     
     ```
     psh "rpm -qa* | grep jazxfbrs
     ```
     
     If a previous version is installed, a result similar to this is displayed. The version number might be different.
     
     ```
     jazxfbrs-9.2-1.9
     ```
     
   - If a SAS 9.3 or SAS 9.4 version of the formats library is currently installed, run this command:
     
     ```
     psh "rpm -qa* | grep acc
     ```
     
     If a previous version is installed, a result similar to this is displayed. The version number might be different.
     
     ```
     accelterafrmt-3.1-1.x86_64
     ```
     
     If the library is not installed on the Teradata nodes, no output is displayed. You can continue with the installation steps in “Installing the SAS Formats Library and Binary Files” on page 98.

2. If a version of the SAS formats library is being installed that has a name that is different from the library that was previously installed, then follow these steps. Here is an example:

   ```
   accelterafrmt-3.1-1 replacing jazxfbrs-9.2-1.6
   ```

   a. Shut down the Teradata database.
      
      ```
      tpareset -y -x shutdown_comment
      ```
This step is required because an older version of the SAS formats library might be loaded in a currently running SAS query.

b. Confirm that the database is shut down.

   pdestate -a

   DOWN/HARDSTOP is displayed if the database is shut down.

c. Remove the old version of the in-database deployment package before you install the updated version.

   • To remove the packages from all nodes concurrently, run this command:

     psh "rpm -e package-name"

     *package-name* is either jazxfbrs.9.version or accelterafmt-version.

     For example, to remove jazxfbrs, run the command psh "rpm -e jazxfbrs-9.2-1.6".

   • To remove the package from each node, run this command on each node:

     rpm -e package-name

     *package-name* is either jazxfbrs.9.version or accelterafmt-version.

3. (Optional) To confirm removal of the package before installing the new package, run this command:

   psh "rpm -q package-name"

   *package-name* is either jazxfbrs.9.version or accelterafmt-version.

   The SAS Formats Library should not appear on any node.

4. Continue with the installation steps in “Installing the SAS Formats Library and Binary Files” on page 98.

---

**Installing the SAS Formats Library and Binary Files**

**Moving the SAS Formats Library and Binary File Package to the Server Machine**

1. Locate the SAS Formats Library for Teradata deployment package file, accelterafmt-3.1-n.x86_64.rpm. *n* is a number that indicates the latest version of the file. If this is the initial installation, *n* has a value of 1.

   The accelterafmt-3.1-n.x86_64.rpm file is located in the *YourSASHome/SASFomatsLibraryforTeradata/3.1/TeradataonLinux/* directory.

   **Note:** The SAS formats library is not required by the SAS Data Quality Accelerator for Teradata.

2. Move the package file to your Teradata database server in a location where it is both Read and Write accessible. You need to move this package file to the server machine in accordance with procedures used at your site. Here is an example using secure copy.

   scp accelterafmt-3.1-n.x86_64.rpm root@teramach1:/sasdir/18NOV16

   This package file is readable by the Teradata Parallel Upgrade Tool.
If you have already deployed the SAS Embedded Process, it is recommended, but not required, that you move the Teradata in-database deployment package to the same location as the SAS Embedded Process deployment package.

**Installing the SAS Formats Library and Binary Files with the Teradata Parallel Upgrade Tool**

This installation should be performed by a Teradata systems administrator in collaboration with Teradata Customer Services. A Teradata Change Control is required when a package is added to the Teradata server. Teradata Customer Services has developed change control procedures for installing the SAS in-database deployment package.

The steps assume full knowledge of the Teradata Parallel Upgrade Tool and your environment. For more information about using the Teradata Parallel Upgrade Tool, see the *Parallel Upgrade Tool (PUT) Reference* which is at the Teradata Online Publications site, located at [http://www.info.teradata.com/GenSrch/eOnLine-Srch.cfm](http://www.info.teradata.com/GenSrch/eOnLine-Srch.cfm). On this page, search for “Parallel Upgrade Tool” and download the appropriate document for your system.

The following steps explain the basic steps to install the SAS formats library package by using the Teradata Parallel Upgrade Tool.

**Note:** The Teradata Parallel Upgrade Tool prompts are subject to change as Teradata enhances its software.

1. Locate the SAS Formats Library and binary file package on your server machine. It must be in a location where it can be accessed from at least one of the Teradata nodes. For more information, see “Moving the SAS Formats Library and Binary File Package to the Server Machine” on page 98.

2. Start the Teradata Parallel Upgrade Tool.

3. Be sure to select all Teradata TPA nodes for installation, including Hot Stand-By nodes.

4. If Teradata Version Migration and Fallback (VM&F) is installed, you might be prompted whether to use VM&F or not. If you are prompted, choose Non-VM&F installation.

5. If the installation is successful, `accelterfmt-3.1-n` is displayed. `n` is a number that indicates the latest version of the file.

   Alternatively, you can manually verify that the installation is successful by running this command from the shell prompt:

   ```
   psh *rpm -q -a* | grep accelterafmt
   ```

---

**Controlling the SAS Embedded Process**

The SAS Embedded Process starts when a query is submitted. The SAS Embedded Process continues to run until it is manually stopped or the database is shutdown. You might want to disable or shutdown the SAS Embedded Process without shutting down the database.

The following commands control the SAS Embedded Process.
### Action performed | Command (by Teradata version)
--- | ---
Provides the status of the SAS Embedded Process. | CALL SQLJ.SERVERCONTROL ('SAS', 'status', :A); *
Shuts down the SAS Embedded Process. | CALL SQLJ.SERVERCONTROL ('SAS', 'shutdown', :A); *
*Note: You cannot shut down until all queries are complete.*
Stops new queries from being started. Queries that are currently running continue to run until they are complete. | CALL SQLJ.SERVERCONTROL ('SAS', 'disable', :A); *
Enables new queries to start running. | CALL SQLJ.SERVERCONTROL ('SAS', 'enable', :A); *

* Note that the Languagename parameter, 'SAS', is required and must be uppercase. The Cmd parameter (for example, 'status'), must be lowercase.

Here is the sequence of operations to stop and then restart the SAS Embedded Process:

1. Disable the SAS Embedded Process to stop new queries from being started.
   
   CALL SQLJ.SERVERCONTROL ('SAS', 'disable', :A);

2. Query the status of the SAS Embedded Process until the status returns this message:
   Hybrid Server is disabled with no UDFs running.
   
   CALL SQLJ.SERVERCONTROL ('SAS', 'status', :A);

   
   CALL SQLJ.SERVERCONTROL ('SAS', 'shutdown', :A);

4. Perform maintenance on the SAS Embedded Process, for example, install a hot fix or upgrade to a new version.

5. Enable the SAS Embedded Process.
   
   CALL SQLJ.SERVERCONTROL ('SAS', 'enable', :A);

6. Test the SAS Embedded Process. The SAS Embedded Process will start when the next SAS query that uses the SAS Embedded Process is sent to the database.

### Teradata Permissions for Publishing Formats and Scoring Models

Because functions are associated with a database, the functions inherit the access rights of that database. It might be useful to create a separate shared database for the SAS scoring functions or the SAS_PUT( ) function so that access rights can be customized as needed.

You must grant the following permissions to any user who runs the scoring or format publishing macros:

- CREATE FUNCTION ON `database` TO `userid`
- DROP FUNCTION ON `database` TO `userid`
- EXECUTE FUNCTION ON `database` TO `userid`
ALTER FUNCTION ON database TO userid

If you use the SAS Embedded Process to run your scoring model, you must grant the following permissions:

SELECT, CREATE TABLE, INSERT ON database TO userid
EXECUTE PROCEDURE ON SAS_SYSFNLIB TO userid
EXECUTE FUNCTION ON SAS_SYSFNLIB TO userid
EXECUTE FUNCTION ON SYSLIB.MonitorVirtualConfig TO userid

Note: If you plan to use SAS Model Manager with the SAS Scoring Accelerator for in-database scoring, additional permissions are required. For more information, see Chapter 12, “Configuring SAS Model Manager,” on page 115.

Documentation for Using In-Database Processing in Teradata

• SAS Data Quality Accelerator for Teradata: User's Guide
• SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide
• SAS In-Database Products: User’s Guide
• SAS Intelligence Platform: Data Administration Guide
• SAS Model Manager: User’s Guide
# Chapter 11
## SAS Data Quality Accelerator for Teradata

### Introduction

Before you use the SAS data cleansing functionality inside the Teradata database, you must perform the following tasks:

- deploy the SAS Embedded Process in the Teradata database (if it has not already been deployed)
- install SAS data quality stored procedures in the Teradata database
- deploy a SAS Quality Knowledge Base (QKB) in the Teradata database

The SAS Embedded Process deployment package delivers shell scripts that enable you to install and manage the SAS data quality stored procedures within the Teradata database. In addition, it contains a shell script that enables you to package the QKB for deployment inside the Teradata database. The QKB is a collection of files that store data and logic that support data management operations. SAS software products reference the QKB when performing data management operations on your data.

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>103</td>
</tr>
<tr>
<td>Deploying the SAS Embedded Process</td>
<td>104</td>
</tr>
<tr>
<td>Installing SAS Data Quality Accelerator Stored Procedures in the Teradata Database</td>
<td>104</td>
</tr>
<tr>
<td>Overview</td>
<td>104</td>
</tr>
<tr>
<td>Using the dq_install.sh Script</td>
<td>105</td>
</tr>
<tr>
<td>Using the dq_grant.sh Script</td>
<td>105</td>
</tr>
<tr>
<td>Obtaining and Deploying a QKB</td>
<td>106</td>
</tr>
<tr>
<td>Obtaining a QKB</td>
<td>106</td>
</tr>
<tr>
<td>Packaging the QKB</td>
<td>106</td>
</tr>
<tr>
<td>Installing the QKB Package File with the Teradata Parallel Upgrade Tool</td>
<td>108</td>
</tr>
<tr>
<td>Validating the Accelerator Installation</td>
<td>108</td>
</tr>
<tr>
<td>Troubleshooting the Accelerator Installation</td>
<td>109</td>
</tr>
<tr>
<td>Updating and Customizing a QKB</td>
<td>111</td>
</tr>
<tr>
<td>Removing SAS Data Quality Accelerator from the Teradata Database</td>
<td>111</td>
</tr>
<tr>
<td>Overview</td>
<td>111</td>
</tr>
<tr>
<td>Using the dq_uninstall.sh Script</td>
<td>112</td>
</tr>
</tbody>
</table>
Each Teradata node needs approximately 8 GB for the QKB.

---

### Deploying the SAS Embedded Process


*Note:* If you have a previous version of the SAS Embedded Process and SAS Data Quality Accelerator for Teradata stored procedures installed in your Teradata database, you must remove them before installing the new version of the SAS Embedded Process. The SAS Data Quality Accelerator for Teradata stored procedures must be removed from the Teradata database before you remove the SAS Embedded Process. For more information, see “Removing SAS Data Quality Accelerator from the Teradata Database” on page 111. Then see “Upgrading or Uninstalling the SAS Embedded Process for Teradata” in SAS Embedded Process: Deployment Guide.

---

### Installing SAS Data Quality Accelerator Stored Procedures in the Teradata Database

#### Overview

The SAS Embedded Process deployment package for Teradata (sepcoretera) installs three scripts in the `/opt/SAS/ep/home/install/pgm` directory of the Teradata database server:

- a stored procedure creation script named `dq_install.sh`
- a user authorization script named `dq_grant.sh`
- a stored procedure removal script named `dq_uninstall.sh`

You will need to run the `dq_install.sh` script to create the data quality stored procedures in the Teradata database and the `dq_grant.sh` script to grant users permission to execute the data quality stored procedures.

The `dq_uninstall.sh` script is provided to enable you to remove the data quality stored procedures from the database. You must remove any data quality stored procedures that have already been installed from the Teradata database before upgrading or re-installing either SAS Data Quality Accelerator for Teradata or the SAS Embedded Process.

*Note:* All three scripts must be run as the root user.
Using the dq_install.sh Script

The dq_install.sh shell script is provided to enable the Teradata system administrator to create the data quality stored procedures in the Teradata database. The dq_install.sh script is located in the /opt/SAS/ep/home/install/pgm directory of the Teradata database server.

The dq_install.sh script requires modification before it can be run. The Teradata administrator must edit the shell script to specify the site-specific Teradata server name and DBC user logon credentials for the DBC_PASS=, DBC_SRVR=, and DBC_USER= variables.

Running dq_install.sh puts the data quality stored procedures into the SAS_SYSFNLIB database and enables the accelerator functionality.

Here is the syntax for executing dq_install.sh:

```bash
./dq_install.sh <-l log-path
```

**log-path**

specifies an alternative name and location for the dq_install.sh log. When this parameter is omitted, the script creates a file named dq_install.log in the current directory.

The next step in the installation is to grant users permission to execute the stored procedures.

Using the dq_grant.sh Script

The dq_grant.sh shell script is provided to enable the Teradata system administrator to grant users authorization to the data quality stored procedures. The dq_grant.sh script is located in the /opt/SAS/ep/home/install/pgm directory of the Teradata database server.

The dq_grant.sh script requires modification before it can be run. The Teradata administrator must edit the shell script to specify the site-specific Teradata server name and DBC user logon credentials for the DBC_SRVR=, DBC_USER=, and DBC_PASS= variables.

Here is the syntax for executing dq_grant.sh:

```bash
./dq_grant.sh <-l log-path> user-name
```

**log-path**

specifies an alternative name and location for the dq_grant.sh log. When this parameter is omitted, the script creates a file named dq_grant.log in the current directory.

**user-name**

is the user name to which permission is being granted. The target user account must already exist in the Teradata database.

The authorizations granted by dq_grant.sh augment existing authorizations that the target user account already has in the Teradata database.

You can verify that authorization was granted successfully for a user by logging on to the database as the user and issuing the following command in BTEQ:

```
call sas_sysfnlib.dq_debug();
```
The command will fail if the user does not have permission. Otherwise, it will have no effect.

The data quality stored procedures are not yet ready to use. A QKB must be installed in the Teradata database for the data quality stored procedures to be usable.

---

**Obtaining and Deploying a QKB**

**Obtaining a QKB**

You can obtain a QKB in one of the following ways:

- Run the SAS Deployment Wizard. In the Select Products to Install dialog box, select the check box for the SAS Quality Knowledge Base for your order. For step-by-step guidance on installing a QKB using the SAS Deployment Wizard, see the *SAS Quality Knowledge Base for Contact Information: Installation and Configuration Guide* or the *SAS Quality Knowledge Base for Product Data: Installation and Configuration Guide*, as appropriate, on the [Quality Knowledge Base for SAS and DataFlux Documentation site](https://support.sas.com/documentation/onlinedoc/).

- Download a QKB from the [SAS Downloads site](https://support.sas.com). Contact your SAS Administrator to determine which QKB has been licensed before downloading.

  Select a QKB, and then follow the installation instructions in the Readme file for your operating environment. To open the Readme file, you must have a SAS profile. When prompted, you can log on or create a new profile.

- Copy a QKB that you already use with other SAS software in your enterprise. Contact your system administrator for its location.

**Packaging the QKB**

Before a QKB can be deployed in the Teradata database, you must package it into an .rpm file. An .rpm file is a file that is suitable for installation on Linux systems that use RPM package management software. SAS Data Quality Accelerator for Teradata provides the `qkb_pack` script to package the QKB into an .rpm file.

Windows and UNIX versions of `qkb_pack` are available. You must run the version that is appropriate for the host environment in which your QKB is installed.

`qkb_pack` is created in the following directories by the SAS Deployment Wizard:

**Windows**

- `C:\Program Files\<SASHome>\SASDataQualityAcceleratorforTeradata\9.4\dqacctera\sasmisc`

**UNIX**

- `<SASHome>\SASDataQualityAcceleratorforTeradata\9.4\install\pgm`

You must execute `qkb_pack` from these locations.

Here is the syntax for executing `qkb_pack`:

**Windows**

```
qkb_pack.cmd qkb-dir out-dir
```
UNIX:

../qkb_pack.sh qkb-dir out-dir

**qkb-dir**

specify the path to the QKB. Use the name of the QKB’s root directory. Typically, the root directory is found at the following directories:

**Windows:** \C:\ProgramData\SAS\QKB\product-identifier\product-version (for example, \C:\ProgramData\SAS\QKB\CI\29 or \C:\ProgramData\SAS\QKB\PD\5).

**UNIX:** /opt/sas/qkb/share

**Note:** On Windows systems, QKB information exists in two locations: in \C:\ProgramData and in \C:\Program Files. For the qkb_pack command, you must specify the \C:\ProgramData location.

**out-dir**

specify the directory where you want the package file to be created.

Here is an example of a command that you might execute to package a SAS QKB for Contact Information that resides on a Windows computer.

cd c:\Program Files\SASHome\SASDataQualityAcceleratorforTeradata\9.4\dqaccter\sasmisc
qkb_pack.cmd c:\ProgramData\SAS\QKB\CI\29 c:\temp\

The default location for the qkb-dir argument is: \C:\ProgramData\SAS\SASQualityKnowledgeBase\CI.

The package file that is created in \C:\temp\ will have a name in the following form:

sasqkb_product-version-timestamp.noarch.rpm

**product**

is a two-character product code for the QKB, such as CI (for Contact Information) or PD (for Product Data).

**version**

is the version number of the QKB.

**timestamp**

is a UNIX datetime value that indicates when qkb_pack was invoked. A UNIX datetime value is stored as the number of seconds since January 1, 1970.

**noarch**

indicates that the package file is platform-independent.

Here is an example of an output file name representing the QKB for Contact Information 29:

sasqkb_ci-29.0-1474057340608.noarch.rpm

After running qkb_pack, put the sasqkb package file on your Teradata database server in a location where it is available for both reading and writing. The package file must be readable by the Teradata Parallel Upgrade Tool. You need to move this package file to the server machine in accordance with procedures used at your site.

Follow the steps in “Installing the QKB Package File with the Teradata Parallel Upgrade Tool” on page 108 to deploy the QKB package file in the Teradata database.
Installing the QKB Package File with the Teradata Parallel Upgrade Tool

This installation should be performed by a Teradata systems administrator in collaboration with Teradata Customer Services. A Teradata Change Control is required when a package is added to the Teradata server. Teradata Customer Services has developed change control procedures for installing the SAS in-database deployment package.

The steps assume full knowledge of the Teradata Parallel Upgrade Tool and your environment. For more information about using the Teradata Parallel Upgrade Tool, see the Parallel Upgrade Tool (PUT) Reference, which is on the Teradata Online Publications site located at [http://www.info.teradata.com/GenSrch/eOnLine-Srch.cfm](http://www.info.teradata.com/GenSrch/eOnLine-Srch.cfm). On this page, search for “Parallel Upgrade Tool” and download the appropriate document for your system.

The following section explains the basic steps to install the sasqkb package file using the Teradata Parallel Upgrade Tool.

**Note:** It is not necessary to stop and restart the Teradata database when you install a QKB. However, if the SAS Embedded Process is running, you must stop it and then re-start it after the QKB is installed. It is also necessary to stop and restart the SAS Embedded Process for QKB updates. See “Controlling the SAS Embedded Process” on page 99 for information about stopping and restarting the SAS Embedded Process.

**Note:** The Teradata Parallel Upgrade Tool prompts are subject to change as Teradata enhances its software.

1. Start the Teradata Parallel Upgrade Tool.
2. Be sure to select all Teradata TPA nodes for installation, including Hot Stand-By nodes.
3. If Teradata Version Migration and Fallback (VM&F) is installed, you might be prompted whether to use VM&F. If you are prompted, choose Non-VM&F installation.

You can verify that the QKB installation was successful by running the following command from the shell prompt on one of the Teradata nodes.

```
psh "rpm -q -a" | grep sasqkb
```

If the installation was successful, the command returns the version number of the sasqkb package. Failure to return an output indicates that a library of that name could not be found.

The QKB is installed in the `/opt/qkb/default` directory of each Teradata node. You are now ready to validate the data quality stored procedures for use.

---

Validating the Accelerator Installation

Here is a simple BTEQ program that can be used to verify that the SAS Data Quality Accelerator for Teradata is operational.

The code first lists the locales that are installed in the QKB. Then it creates a table named Dqacceltest and executes the DQ_GENDER() stored procedure on the table.
Before running the example, substitute a real value for the `output_table_1`, `output_table_2`, and `locale` variables throughout the program. For `locale`, use one of the values returned by the DQ_LIST_LOCALES() stored procedure. This example assumes that the SAS Data Quality Accelerator for Teradata is using the QKB for Contact Information.

The example also sets the SAS Data Quality Accelerator DQ_OVERWRITE_TABLE option to create temporary output tables in the SAS Data Quality Accelerator session. If you run the example again in the same SAS Data Quality Accelerator session, the new output tables overwrite any existing output tables and the output tables are automatically discarded at the end of the session. The DROP TABLE statement removes table Dqacceltest from your database.

```sql
call sas_sysfnlib.dq_list_locales('mydb.output_table_1');
select * from mydb.output_table_1;

call sas_sysfnlib.dq_set_option('DQ_OVERWRITE_TABLE', '1');

create table mydb.dqacceltest (id_num integer, name varchar(64))
    unique primary index(id_num);

insert into mydb.dqacceltest (id_num, name) values (1, 'John Smith');
insert into mydb.dqacceltest (id_num, name) values (2, 'Mary Jones');

call sas_sysfnlib.dq_gender('Name', 'mydb.dqacceltest', 'name', 'id_num',
    'mydb.output_table_2', 'locale');

select gender from mydb.output_table_2;
drop table mydb.dqacceltest;
```

If the request was successful, the SELECT statement produces an output table that contains this:

<table>
<thead>
<tr>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
</tr>
<tr>
<td>F</td>
</tr>
</tbody>
</table>

**Troubleshooting the Accelerator Installation**

Q. I ran the sample code and the output tables were not created in my user schema. What now?

A. The stored procedures can fail if one or more of the following conditions are true:

- The request specifies an output location to which the user does not have Write permission. Verify that you have access to the database that is specified in the `output_table` parameters.
- The data quality stored procedures are not installed correctly. Verify that the stored procedures are in the SAS_SYSFNLIB database by executing the following command in BTEQ:

```sql
select TableName from dbc.tables where databasename='SAS_SYSFNLIB' and tablename like 'dq_%';
```
The command should return a list similar to the following list (This is not a complete list.):

<table>
<thead>
<tr>
<th>TableName</th>
</tr>
</thead>
<tbody>
<tr>
<td>dq_set_qkb</td>
</tr>
<tr>
<td>dq_match_parsed</td>
</tr>
<tr>
<td>dq_i_drop_view_if_exists</td>
</tr>
<tr>
<td>dq_i_get_option_default</td>
</tr>
<tr>
<td>dq_debug</td>
</tr>
<tr>
<td>dq_propercase</td>
</tr>
<tr>
<td>dq_i_tbl_dbname</td>
</tr>
<tr>
<td>dq_i_drop_tbl_if_exists</td>
</tr>
<tr>
<td>dq_set_option</td>
</tr>
<tr>
<td>dq_error</td>
</tr>
<tr>
<td>dq_standardize</td>
</tr>
<tr>
<td>dq_standardize_parsed</td>
</tr>
<tr>
<td>dq_debug2</td>
</tr>
<tr>
<td>dq_i_invoke_table</td>
</tr>
<tr>
<td>dq_lowercase</td>
</tr>
<tr>
<td>dq_set_locale</td>
</tr>
<tr>
<td>dq_extract</td>
</tr>
<tr>
<td>dq_uppercase</td>
</tr>
<tr>
<td>dq_list_bindings</td>
</tr>
<tr>
<td>dq_replace_tags</td>
</tr>
<tr>
<td>dq_list_defns</td>
</tr>
<tr>
<td>dq_i_call_dp</td>
</tr>
<tr>
<td>dq_i_get_bool_option</td>
</tr>
<tr>
<td>dq_i_gen_toktxt</td>
</tr>
<tr>
<td>dqt_codegen</td>
</tr>
<tr>
<td>dq_match</td>
</tr>
<tr>
<td>dq_parse</td>
</tr>
<tr>
<td>dqt_trace</td>
</tr>
<tr>
<td>dq_pattern</td>
</tr>
<tr>
<td>dq_i_clear_tok_tibs</td>
</tr>
<tr>
<td>dqt_tokname_tmp</td>
</tr>
<tr>
<td>dq_format</td>
</tr>
<tr>
<td>dq_list_locales</td>
</tr>
<tr>
<td>dq_i_invoke_scalar</td>
</tr>
<tr>
<td>dq_i_invoke_preparsed</td>
</tr>
<tr>
<td>dq_bind_token</td>
</tr>
<tr>
<td>dq_gender</td>
</tr>
</tbody>
</table>

If the procedures are absent, run the dq_install.sh script again, making sure that you are logged in as Teradata system administrator.

- Permission to the data quality stored procedures is not granted correctly. Verify that the target user name submitted to the dq_grant.sh script is a valid user account in the Teradata database. Verify that the database server and granter information in the dq_grant.sh shell script is correct.

- The QKB is not in the correct location. Look for subdirectories similar to the following in the /opt/qkb/default directory on the Teradata nodes: chopinfo, grammar, locale, phonetx, regexlib, scheme, and vocab.

- Your SQL request does not use the Teradata dialect. The stored procedures are invoked with the CALL keyword from any product that supports the Teradata SQL dialect. When you submit the data quality stored procedures in the SAS SQL
procedure using explicit pass-through, the database connection is made in ANSI mode by default. You must specify the MODE= option to switch to Teradata mode. Consult the SAS/ACCESS Interface to Teradata documentation for more information about the MODE= option. Consult appropriate documentation for how to set Teradata mode in other client programs.

---

### Updating and Customizing a QKB

SAS provides regular updates to the QKB. It is recommended that you update your QKB each time a new one is released. For a listing of the latest enhancements to the QKB, see “What’s New in SAS Quality Knowledge Base.” The What’s New document is available on the Quality Knowledge Base (QKB) for SAS and DataFlux Documentation site on support.sas.com.

Check the What’s New for each QKB to determine which definitions have been added, modified, or deprecated, and to learn about new locales that might be supported. Contact your SAS software representative to order updated QKBS and locales. To deploy a new QKB, follow the steps in “Packaging the QKB” on page 106 and “Installing the QKB Package File with the Teradata Parallel Upgrade Tool” on page 108. The accelerator supports one QKB in the Teradata database.

The standard definitions in the QKB are sufficient for performing most data quality operations. However, you can use the Customize feature of DataFlux Data Management Studio to modify the QKB definitions to meet specific needs.

If you want to customize your QKB, then as a best practice, SAS recommends that you customize your QKB on a local workstation before copying it to the Teradata database for deployment. When updates to the QKB are required, merge your customizations into an updated QKB locally, and copy the updated, customized QKB to the Teradata node. This enables you to deploy a customized QKB to the Teradata database using the same steps that you would use to deploy a standard QKB. Copying your customized QKB from a local workstation into your cluster also means that you will have a backup of the QKB on your local workstation. 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.

---

### Removing SAS Data Quality Accelerator from the Teradata Database

#### Overview

Before you can upgrade, re-install, or permanently remove SAS Data Quality Accelerator for Teradata or the SAS Embedded Process, you must remove any existing data quality stored procedures from the Teradata database. The stored procedures are removed from the Teradata database by using the dq_uninstall.sh script. For more information about this script, see “Using the dq_uninstall.sh Script” on page 112.

It is not necessary to remove the QKB when upgrading or re-installing software. QKB deployment steps automatically overwrite an older version of the QKB when you install a new one. For information to replace the QKB, see “Updating and Customizing a QKB” on page 111 and “Obtaining a QKB” on page 106.
When permanently removing SAS Data Quality Accelerator for Teradata from the Teradata database server, follow whatever procedure is appropriate at your site for removing the QKB. The Teradata administrator also needs to remove data quality authorizations from the Teradata database in accordance with site procedures.

**Using the dq_uninstall.sh Script**

*Note:* Stop the embedded process by using the instructions at “Controlling the SAS Embedded Process” on page 99 before following these steps. Stopping the SAS Embedded Process ensures that none of the accelerator files are locked when dq_uninstall.sh attempts to remove them.

The accelerator provides the dq_uninstall.sh shell script for removing the data quality stored procedures from the Teradata database. The dq_uninstall.sh script is located in the `/opt/SAS/ep/home/install/pgm` directory of the Teradata database server.

The dq_uninstall.sh script requires modification before it can be run. The Teradata administrator must edit the shell script to specify the site-specific Teradata server name and DBC user logon credentials for the DBC_PASS=, DBC_SRVR=, and DBC_USER= variables.

Here is the syntax for executing dq_uninstall.sh:

```
./dq_uninstall.sh <-l log-path>
```

*log-path*

specifies an alternative name and location for the dq_uninstall.sh log. When this parameter is omitted, the script creates a file named dq_uninstall.log in the current directory.

Running dq_uninstall.sh disables the SAS Data Quality Accelerator for Teradata functionality and removes the data quality stored procedures from the database.
Part 10

Configurations for SAS Model Manager

Chapter 12
Configuring SAS Model Manager
Chapter 12
Configuring SAS Model Manager

Preparing a Data Management System for Use with SAS Model Manager . . . . . . 115
Prerequisites ................................................................. 115
Overview of Preparing a Data Management System for Use with SAS Model Manager .................................. 116
Configuring a Database ...................................................... 116
SAS Embedded Process Publish Method ................................ 116
Scoring Function Publish Method ......................................... 117
Finding the JDBC JAR Files ................................................. 118
Configuring a Hadoop Distributed File System ......................... 119

Preparing a Data Management System for Use with SAS Model Manager

Prerequisites

SAS Foundation, SAS/ACCESS, and the in-database deployment package for the database must be installed and configured before you can prepare a data management system (database or file system) for use with SAS Model Manager. For more information, see the chapter for your type of database or file system in this guide. Here are the databases and file systems that can be used with SAS Model Manager:

- DB2
- Greenplum
- Hadoop
- Netezza
- Oracle
- SAP HANA
- Teradata
Overview of Preparing a Data Management System for Use with SAS Model Manager

Additional configuration steps are required to prepare a data management system (database or file system) for publishing and scoring in SAS Model Manager if you plan to use the scoring function (MECHANISM=STATIC) publish method or the SAS Embedded Process (MECHANISM=EP) publish method. If you want to store the scoring function metadata tables in the database, then the SAS Model Manager In-Database Scoring Scripts product must be installed before the database administrator (DBA) can prepare a database for use with SAS Model Manager.

During the installation and configuration of SAS 9.4 products, the SAS Model Manager In-Database Scoring Scripts product is installed on the middle-tier server or another server tier if it is included in the custom plan file.

The location of the SAS installation directory is specified by the user. Here is the default installation location for the SAS Model Manager In-Database Scoring Scripts product on a Microsoft Windows server:

C:\Program Files\SASHome\SASModelManagerInDatabaseScoringScripts

The script installation directory includes a directory that specifies the version of SAS Model Manager (currently 14.2). The files and subdirectories that are needed to prepare a database for use by SAS Model Manager are located in the version directory. The Utilities subdirectory contains two SQL scripts for each type of database: a Create Tables script and a Drop Tables script. The DBA needs these SQL scripts to create the tables needed by the SAS Model Manager to publish scoring functions.

Note: The database tables store SAS Model Manager metadata about scoring functions.

If your system is configured for Kerberos authentication for Hadoop or Teradata, each user must have a valid Kerberos ticket. You must also complete post-installation configuration steps to enable users to publish models from the SAS Model Manager web application. For more information, see SAS Model Manager: Administrator’s Guide.

Configuring a Database

SAS Embedded Process Publish Method

To enable users to publish scoring model files to a database from SAS Model Manager using the SAS Embedded Process, follow these steps:

1. Create a separate database where the tables can be stored.
2. Set the user access permissions for the database.

For information about the permissions that are required for specific databases, see the following topics:

- “DB2 Permissions”
- “Greenplum Permissions”
- “Netezza Permissions”
- “Oracle Permissions”
- “SAP HANA Permissions”
3. SAS Model Manager requires that the following database and table permissions be set at minimum. Verify that the permissions that you have set on your database include these permissions.

   a. GRANT CREATE, DROP, EXECUTE, and ALTER permissions for functions and procedures.

   b. GRANT CREATE and DROP permissions for tables. With these permissions, users can validate the scoring results when publishing scoring model files using SAS Model Manager.

   c. Run the database-specific macro to create a table in the database to store the published model scoring files. The value of the MODELTABLE= argument in the macro should match the specification of the In-Database Options for SAS Model Manager in SAS Management Console. For more information, see In-Database Options.

      If the Use model manager table option is set to No, then the model-table-name should be sas_model_table. Otherwise, it should be sas_mdlmgr_ep.

      Here is an example of the create model table macro for Teradata:

      `%INDTD_CREATE_MODELTABLE(DATABASE=database-name, MODELTABLE=model-table-name, ACTION=CREATE);`

      For more information about creating a table for a specific database, see the SAS In-Database Products: User’s Guide.

Scoring Function Publish Method

To enable users to publish scoring functions to a database from SAS Model Manager, follow these steps:

1. Create a separate database where the tables can be stored.

2. Set the user access permissions for the database.

   For information about the permissions that are required for specific databases, see the following topics:

   - “DB2 Permissions”
   - “Greenplum Permissions”
   - “Netezza Permissions”
   - “Teradata Permissions for Publishing Formats and Scoring Models”

3. SAS Model Manager requires that the following database and table permissions be set at minimum. Verify that the permissions that you have set on your database include these permissions.

   a. GRANT CREATE, DROP, EXECUTE, and ALTER permissions for functions and procedures.

   b. GRANT CREATE and DROP permissions for tables. With these permissions, users can validate the scoring results when publishing a scoring function using SAS Model Manager.

   c. GRANT SELECT, INSERT, UPDATE, and DELETE permissions for SAS Model Manager metadata tables.
d. GRANT SELECT permission for the following views to validate the scoring function names:

- syscat.functions for DB2
- pg_catalog.pg_proc for Greenplum
- dbc.functions for Teradata
- _v_function for Netezza

Note: If scoring input tables, scoring output tables, or views exist in another database, then the user needs appropriate permissions to access those tables or views.

4. Navigate to the `sasinstalldir\SASModelManagerInDatabaseScoringScripts\14.2\Utilities` directory to find the Create Tables and Drop Tables scripts for your database. Then, follow these steps:

a. Verify the statements that are specified in the Create Tables script. Here are the names of the scripts for each type of database:

- DB2 SQL scripts: createTablesDB2.sql and dropTablesDB2.sql
- Greenplum SQL scripts: createTablesGreenplum.sql and dropTablesGreenplum.sql
- Netezza SQL scripts: createTablesNetezza.sql and dropTablesNetezza.sql
- Teradata SQL scripts: createTablesTD.sql and dropTablesTD.sql

b. Execute the Create Tables script for a specific type of database.

5. Download the JDBC driver JAR files and place them in the `\lib` directory on the web application server where the SAS Model Manager web application is deployed.

The default directory paths for the SAS Web Application Server are the following:

- single server install and configuration
  ```
  \sasconfigdir\Lev1\Web\WebAppServer\SASServer1_1\lib
  ```
  This is an example of the directory path: C:\SAS\Config\Lev1\Web\WebAppServer\SASServer1_1\lib

- multiple server install and configuration
  ```
  \sasconfigdir\Lev1\Web\WebAppServer\SASServer11_1\lib
  ```
  This is an example of the directory path: C:\SAS\Config\Lev1\Web\WebAppServer\SASServer11_1\lib

Note: You must have Write permission to place the JDBC driver JAR files in the `\lib` directory. Otherwise, you can have the server administrator download them for you.

For more information, see “Finding the JDBC JAR Files” on page 118.

6. Restart the SAS servers on the web application server.

Finding the JDBC JAR Files

The DB2 JDBC JAR files are `db2jcc.jar` and `db2jcc_license_cu.jar`. The DB2 JDBC JAR files can be found on the server on which the database client was
installed. For example, the default location for Windows is `C:\Program Files\IBM\SQLLIB\java`.

The Greenplum database uses the standard PostgreSQL database drivers. The PostgreSQL JDBC JAR file can be found on the PostgreSQL – JDBC Driver site at https://jdbc.postgresql.org/download.html. An example of a JDBC driver name is `postgresql-9.2-1002.jdbc4.jar`.

The Netezza JDBC JAR file is `nzjdbc.jar`. The Netezza JDBC JAR file can be found on the server on which the database client was installed. For example, the default location for Windows is `C:\JDBC`.

The Teradata JDBC JAR files are `terajdbc4.jar` and `tdgssconfig.jar`. The Teradata JDBC JAR files can be found on the Teradata website at http://www.teradata.com. Select Support → Downloads → Developer Downloads, and then click JDBC Driver in the table.

For more information about the database versions that are supported, see the SAS Foundation system requirements documentation for your operating environment.

### Configuring a Hadoop Distributed File System

To enable users to publish scoring model files to a Hadoop Distributed File System (HDFS) from SAS Model Manager using the SAS Embedded Process, follow these steps:

1. Create an HDFS directory where the model files can be stored.
   
   *Note:* The path to this directory is used when a user publishes a model from the SAS Model Manager user interface to Hadoop.

2. Grant users Write access permission to the HDFS directory. For more information, see “Hadoop Permissions” on page 53.

3. Add this line of code to the autoexec_usermods.sas file that is located in the Windows directory `\SAS-configuration-directory\Levn\SASApp\WorkspaceServer\`

   ```sas
   %let HADOOP_Auth = Kerberos or blank;
   ```

   **UNIX Specifics**

   The location of the autoexec_usermods.sas file for UNIX is `/SAS-configuration-directory/Levn/SASApp/WorkspaceServer/`.

   If your Hadoop server is configured with Kerberos, set the HADOOP_Auth variable to Kerberos. Otherwise, leave it blank.

4. (Optional) If you want users to be able to copy the publish code and execute it using Base SAS, then this line of code must be added to the sasv9.cfg file that is located in the Windows directory `\SASHome\SASFoundation\9.4\`

   ```sas
   -AUTOEXEC \'\SAS-configuration-directory\Levn\SASApp\WorkspaceServer\autoexec_usermods.sas'
   ```

   **UNIX Specifics**

   The location of the sasv9.cfg file for UNIX is `/SASHome/SASFoundation/9.4/`.

5. (Optional) If your Hadoop distribution is using Kerberos authentication, each user must have a valid Kerberos ticket to access SAS Model Manager. You must also complete additional post-installation configuration steps to enable users to publish
models to a Hadoop Distributed File System (HDFS) from SAS Model Manager. For more information, see *SAS Model Manager: Administrator’s Guide*.
Recommended Reading

Here is the recommended reading list for this title:

- SAS/ACCESS for Relational Databases: Reference
- SAS Contextual Analysis In-Database Scoring for Hadoop: Administrator’s Guide
- SAS Contextual Analysis In-Database Scoring for Hadoop: User’s Guide
- SAS Data Loader for Hadoop: User’s Guide
- SAS Data Quality Accelerator for Teradata: User’s Guide
- SAS DS2 Language Reference
- SAS Embedded Process: Deployment Guide
- SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS
- SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide
- SAS In-Database Products: User’s Guide
- SAS Model Manager: Administrator’s Guide
- SAS Viya Deployment Guide

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
Index

Special Characters
%INDAC_PUBLISH_FORMATS macro 9
%INDAC_PUBLISH_MODEL macro 9
%INDB2_PUBLISH_COMPILEUDF
  macro 18
  running 19
  syntax 21
%INDB2_PUBLISH_DELETEUDF
  macro 22
  running 23
  syntax 24
%INDB2_PUBLISH_FORMATS macro 14
%INDB2_PUBLISH_MODEL macro 14
%INDGP_PUBLISH_COMPILEUDF
  macro 38
  running 39
  syntax 41
%INDGP_PUBLISH_COMPILEUDF_E
  P macro 42
  running 42
  syntax 44
%INDGP_PUBLISH_FORMATS macro 34
%INDGP_PUBLISH_MODEL macro 34
%INDHN_PUBLISH_MODEL macro 89
%INDNZ_PUBLISH_COMPILEUDF
  macro 73
  running 73
  syntax 75
%INDNZ_PUBLISH_FORMATS macro 68
%INDNZ_PUBLISH_JAZLIB macro 70
%INDOR_PUBLISH_MODEL macro 68
%INDOR_PUBLISH_MODEL macro 83
%INDTD_PUBLISH_FORMATS macro 96
%INDTD_PUBLISH_MODEL macro 96

A
ACTION= argument
  %INDB2_PUBLISH_COMPILEUDF
    macro 21
  %INDB2_PUBLISH_DELETEUDF
    macro 24
  %INDGP_PUBLISH_COMPILEUDF
    macro 41
  %INDGP_PUBLISH_COMPILEUDF_E
    P macro 45
  %INDNZ_PUBLISH_COMPILEUDF
    macro 76
  %INDNZ_PUBLISH_JAZLIB macro 72
additional configuration
  Hadoop 55
  Amazon S3
  object storage 61
  Aster
  documentation for publishing formats and scoring models 10
  permissions 10
  prerequisites 9
  SAS Embedded Process 9
  SAS/ACCESS Interface to Aster 9
  authorization for stored procedures 105

B
binary files
  for DB2 functions 16, 17
  for Greenplum functions 37
  for Netezza functions 70
BTEQ 108

C
COMPILER_PATH= argument
  %INDB2_PUBLISH_COMPILEUDF
    macro 21
configuration
  DB2 15
  Greenplum 35
  HDFS for Model Manager 119
Model Manager database 116
Netezza 69
Oracle 84
SAP HANA 90
Teradata 96
controlling the SAS Embedded Process
DB2 27
create users and objects
Oracle 84
customizing the QKB 111

D
data quality stored procedures
See stored procedures
DATABASE= argument
%INDB2_PUBLISH_COMPILEUDF macro 21
%INDB2_PUBLISH_DELETEUDF macro 24
%INDGP_PUBLISH_COMPILEUDF macro 41
%INDGP_PUBLISH_COMPILEUDF_EP macro 45
%INDNZ_PUBLISH_COMPILEUDF macro 75
%INDNZ_PUBLISH_JAZLIB macro 72
DataFlux Data Management Studio
customizing the QKB 111
DB2
controlling the SAS Embedded Process 27
documentation for publishing formats or scoring models 30
function publishing process 14
in-database deployment package 13
installation and configuration 15
JDBC Driver 118
permissions 28
preparing for SAS Model Manager use 115
SAS/ACCESS Interface 13
unpacking self-extracting archive files 17
DB2PATH= argument
%INDB2_PUBLISH_COMPILEUDF macro 21
DB2SET command 18
syntax 26
DBSCHEMA= argument
%INDNZ_PUBLISH_COMPILEUDF macro 75
%INDNZ_PUBLISH_JAZLIB macro 72
documentation
for in-database processing in Hadoop 54
for in-database processing in SAP HANA 92
for publishing formats and scoring models in Aster 10
for publishing formats and scoring models in DB2 30
for publishing formats and scoring models in Greenplum 48
for publishing formats and scoring models in Netezza 79
for publishing formats and scoring models in Teradata 101
for publishing scoring models in Oracle 85
dq_grant.sh script 104, 105
dq_install.sh script 104, 105
dq_uninstall script 104
dq_uninstall.sh script 112

F
formats library
DB2 installation 16
Greenplum installation 37
Netezza installation 70
Teradata installation 98
function publishing process
DB2 14
Netezza 68
functions
SAS_COMPILEUDF (DB2) 16, 18, 25
SAS_COMPILEUDF (Greenplum) 37, 38, 45
SAS_COMPILEUDF (Netezza) 70, 73
SAS_COPYUDF (Greenplum) 45
SAS_DEHEXUDF (Greenplum) 45
SAS_DELETEUDF (DB2) 16, 22, 25
SAS_DIRECTORYUDF (Greenplum) 45
SAS_DIRECTORYUDF (Netezza) 73
SAS_EP (Greenplum) 46
SAS_HEXTOTEXTUDF (Netezza) 73
SAS_PUT( ) (Aster) 9
SAS_PUT( ) (DB2) 14
SAS_PUT( ) (Greenplum) 34
SAS_PUT( ) (Netezza) 68
SAS_PUT( ) (Teradata) 96

G
global variables
See variables
Greenplum
documentation for publishing formats and scoring models 48
in-database deployment package 33
installation and configuration 35
JDBC Driver 119
permissions 48
preparing for SAS Model Manager use 115
SAS Embedded Process 46
SAS/ACCESS Interface 33
semaphore requirements 47
unpacking self-extracting archive files 37

H
Hadoop additional configuration 55
configuring HDFS using Model Manager 119
HCatalog 56
in-database deployment package 51
memory usage 60, 61
performance adjustments 58
permissions 53
preparing for SAS Model Manager use 115
SAS/ACCESS Interface 51
Spark 63
trace level 57
HCatalog
prerequisites 56
SAS client configuration 56
SAS server-side configuration 56

I
in-database deployment package for DB2 overview 14
prerequisites 13
in-database deployment package for Greenplum overview 34
prerequisites 33
in-database deployment package for Hadoop prerequisites 51
in-database deployment package for Netezza overview 68
prerequisites 67
in-database deployment package for Teradata overview 96
prerequisites 95
INDCONN macro variable 20, 23, 40, 43, 71, 74
installation DB2 15
Greenplum 35
Netezza 69
SAS Embedded Process (DB2) 16
SAS formats library 16, 37, 70, 99
scripts 104
Teradata 96
troubleshooting 109
verifying 108

J
JDBC Driver
DB2 118
Greenplum 119
Netezza 119
Teradata 119
JDBC JAR file locations 118

M
macro variables
See variables
macros
%INDAC_PUBLISH_FORMATS 9
%INDAC_PUBLISH_MODEL 9
%INDB2_PUBLISH_COMPILEUDF 19, 21
%INDB2_PUBLISH_DELETEUDF 22, 24
%INDB2_PUBLISH_FORMATS 14
%INDB2_PUBLISH_MODEL 14
%INDGP_PUBLISH_COMPILEUDF 38, 41
%INDGP_PUBLISH_COMPILEUDF_EP 44
%INDGP_PUBLISH_FORMATS 34
%INDGP_PUBLISH_MODEL 34
%INDHN_PUBLISH_MODEL 89
%INDNZ_PUBLISH_COMPILEUDF 73, 75
%INDNZ_PUBLISH_FORMATS 68
%INDNZ_PUBLISH_JAZLIB 70, 72
%INDNZ_PUBLISH_MODEL 68
%INDOR_PUBLISH_MODEL 83
%INDTD_PUBLISH_FORMATS 96
%INDTD_PUBLISH_MODEL 96
MapR
additional configuration 57
jobhistory properties 61
Model Manager
configuration 115
configuring a database 116
configuring HDFS 119
creating tables 117
JDBC Driver 118

N
Netezza
documentation for publishing formats and scoring models 79
function publishing process 68
in-database publishing deployment package 67
JDBC Driver 119
permissions 77
preparing for SAS Model Manager use 115
publishing SAS formats library 70
SAS/ACCESS Interface 67

O
OBJNAME= argument
%INDB2_PUBLISH_COMPILEUDF macro 22
OBJPATH= argument
%INDGP_PUBLISH_COMPILEUDF macro 41
%INDGP_PUBLISH_COMPILEUDF_EP macro 44
Oracle
configuration 84
create users and objects 84
documentation for scoring models 85
permissions 85
preparing for SAS Model Manager use 115
prerequisites 83
SAS Embedded Process 83
SAS/ACCESS Interface 83
OUTDIR= argument
%INDB2_PUBLISH_COMPILEUDF macro 22
%INDB2_PUBLISH_DELETEUDF macro 25
%INDGP_PUBLISH_COMPILEUDF macro 42
%INDGP_PUBLISH_COMPILEUDF_EP macro 45
%INDNZ_PUBLISH_COMPILEUDF macro 76
%INDNZ_PUBLISH_JAZLIB macro 73

P
permissions

Q
QKB
customizing 111
packaging for deployment 106
updates 111
qkb_pack script 106

R
removing stored procedures 112
RPM file (Teradata) 98

S
SAP HANA
configuration 90
documentation for in-database processing 92
permissions 91
prerequisites 89
SAS Embedded Process 89, 90
SAS/ACCESS Interface to SAP HANA 89
semaphore requirements 91
SAS Embedded Process
Aster 9
cHECK status (Teradata) 99
controlling (Greenplum) 46
controlling (SAP HANA) 90
controlling (Teradata) 99
disable or enable (Teradata) 99
Greenplum 46
Hadoop 51, 52
Oracle 83
SAP HANA 89, 90
shutdown (Teradata) 99
SAS Embedded Process (Hadoop)
  additional configuration 55
  HCatalog 56
  memory usage 60, 61
  overview 52
  performance adjustments 58
Spark 63
trace level 57
SAS FILENAME SFTP statement (DB2) 14
SAS formats library
  DB2 16
  Greenplum 37
  Netezza 70
  Teradata 98
upgrading from a previous version
  (Greenplum) 35
upgrading from a previous versions
  (DB2) 16
upgrading from a previous versions
  (Teradata) 97
SAS Formats Library
  Teradata 96
SAS Foundation 9, 51, 67, 83, 89, 95
SAS In-Database products 4
SAS_COMPILEUDF function
  actions for DB2 18
  actions for Greenplum 38
  actions for Netezza 73
  binary files for DB2 16
  binary files for Greenplum 37
  binary files for Netezza 70
  validating publication for DB2 25
  validating publication for Greenplum 45
SAS_COPYUDF function 38
  validating publication for Greenplum 45
SAS_DEHEXUDF function 39
  validating publication for Greenplum 45
SAS_DELETEUDF function
  actions for DB2 22
  binary files for DB2 16
  validating publication for DB2 25
SAS_DIRECTORYUDF function 38, 73
  validating publication for Greenplum 45
SAS_EP function
validating publication for Greenplum 46
SAS_HEXITOTEXTUDF function 73
SAS_PUT( ) function
  Aster 9
  DB2 14
  Greenplum 34
  Netezza 68
  Teradata 96
SAS/ACCESS Interface to Aster 9
SAS/ACCESS Interface to DB2 13
SAS/ACCESS Interface to Greenplum 33
SAS/ACCESS Interface to Hadoop 51
SAS/ACCESS Interface to Netezza 67
SAS/ACCESS Interface to Oracle 83
SAS/ACCESS Interface to SAP HANA 89
SAS/ACCESS Interface to Teradata 95
SASLIB database (Netezza) 73
SASLIB schema
  DB2 19, 23
  Greenplum 39
SASUDF_COMPILER_PATH global variable 19
SASUDF_DB2PATH global variable 19
scoring functions in SAS Model Manager 116
scripts for installation 104
self-extracting archive files
  unpacking for DB2 16, 17
  unpacking for Greenplum 37
semaphore requirements
  Greenplum 47
  SAP HANA 91
SFTP statement 14
SSH software (DB2) 14
stored procedures
  creating 105
  removing from database 112
T
tables
  creating for SAS Model Manager 117
Teradata
  BTEQ 108
documentation for publishing formats
  and scoring models 101
in-database deployment package 95
installation and configuration 96
JDBC Driver 119
Parallel Upgrade Tool 99
permissions 100
preparing for SAS Model Manager use 115
SAS Formats Library 96
SAS/ACCESS Interface 95
Teradata Parallel Upgrade Tool
using to deploy the Teradata in-database deployment package 96
troubleshooting installation 109

U
unpacking self-extracting archive files
for DB2 16, 17
for Greenplum 37
upgrading from a previous version
DB2 16
Greenplum 35
Teradata 97
user authorization for stored procedures
105

V
validating publication of functions and variables for DB2 25
validating publication of functions for Greenplum 45, 46
variables
INDCONN macro variable 20, 23, 40, 43, 71, 74
SASUDF_COMPILER_PATH global variable 19
SASUDF_DB2PATH global variable 19
verifying installation 108
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.