THE POWER TO KNOW.

SAS® 9.4 In-Database Products

Administrator's Guide

Fifth Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2015. *SAS® 9.4 In-Database Products: Administrator's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.4 In-Database Products: Administrator's Guide, Fifth Edition

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

February 2015

Contents

Chapter 1
Introduction to the Administrator's Guide

Overview of SAS In-Database Products

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

Note: The SAS Scoring Accelerator for SPD Server does not require any additional installation or configuration.

What Is Covered in This Document?

This document provides detailed instructions for installing and configuring the components that are needed for in-database processing using the SAS/ACCESS Interface, the SAS Scoring Accelerator, the SAS Analytics Accelerator, the SAS Data Loader for Hadoop, the SAS Data Quality Accelerator for Teradata, and the In-Database Code Accelerator. These components are contained in a deployment package that is specific for your database.

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

- SAS Embedded Process for Aster 9.4
- SAS Formats Library for DB2 3.1
- SAS Embedded Process for DB2 9.4
- SAS Formats Library for Greenplum 3.1
- SAS Embedded Process for Greenplum 9.4
- SAS Embedded Process for Hadoop 9.4
- SAS Formats Library for Netezza 3.1
- SAS Embedded Process for Oracle 9.4
- SAS Embedded Process for SAP HANA 9.4
- SAS Formats Library for Teradata 3.1
- SAS Embedded Process for Teradata 9.4

If you want to 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.

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.

Note: Administrative tasks for the SAS Analytics Accelerator are currently in the *SAS Analytics Accelerator for Teradata: User's Guide*.

Chapter 2
Administrator’s Guide for Aster

In-Database Deployment Package for Aster

Prerequisites

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

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

Overview of the In-Database Deployment Package for Aster

This section describes how to install and configure the in-database deployment package for Aster (SAS Embedded Process).

The in-database deployment package 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*.

The in-database deployment package for Aster includes 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.

Aster Installation and Configuration

Aster Installation and Configuration Steps

1. If you are upgrading from or reinstalling a previous release, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 4 before installing the in-database deployment package.
2. Install the in-database deployment package.

   For more information, see "Installing the In-Database Deployment Package Binary Files for Aster" on page 4.

Upgrading from or Reinstalling a Previous Version

Follow these steps to upgrade from or reinstall a previous release.

1. Log on to the queen node.

   ```
   ssh -l root name-or-ip-of-queen-node
   ```

2. Move to the partner directory.

   ```
   cd /home/beehive/partner
   ```

3. If a SAS directory exists in the partner directory, enter this command to remove an existing installation from the queen.

   ```
   rm -rf SAS
   ```

   If you want to perform a clean install, enter these commands to remove the SAS directory from all the workers.

   ```
   location=/home/beehive/partner/SAS/
   for ip in `cat /home/beehive/cluster-management/hosts | grep node |
      awk '{print $3}'`; \
   do \
      echo $ip; \
      ssh $ip "rm -r $location"; \
   done
   rm -rf $location;
   ```

Installing the In-Database Deployment Package Binary Files for Aster

The in-database deployment package binary files for Aster are contained in a self-extracting archive file named tkindbsrv-9.4_M2-*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 reinstall or upgrade, *n* is incremented by 1. The self-extracting archive file is located in the ***`SAS-installation-directory`*****`/SASTKInDatabaseServer/9.4/AsternClusteronLinuxx64/`** directory.

To install the in-database deployment package binary files for Aster, you need root privileges for the queen node. Once you are logged in to the queen node as root, you need to create a directory in which to put tkindbsrv-9.4_M2-*n*_lax.sh, execute tkindbsrv-9.4_M2-*n*_lax.sh, and install the SAS_SCORE( ) and the SAS_PUT( ) SQL/MR functions.

Enter these commands to install the SAS System Libraries and the binary files:

1. Change the directory to the location of the self-extracting archive file.

   ```
   cd SAS-installation-directory/SASTKInDatabaseServer/9.4/AsternClusteronLinuxx64/
   ```

2. Log on to the queen node.

   ```
   ssh -l root name-or-ip-of-queen-node
   ```

3. Move to the parent of the partner directory.

   ```
   cd /home/beehive/
   ```

4. Create a partner directory if it does not already exist.

   ```
   mkdir partner
   ```

5. Move to the partner directory.

   ```
   cd partner
   ```

6. From the SAS client machine, use Secure File Transfer Protocol (SFTP) to transfer the self-extracting archive file to the partner directory.

   a. Using a method of your choice, start the SFTP client.

      Here is an example of starting SFTP from a command line.

      ```
      sftp root@name-or-ip-of-queen-node:/home/beehive/partner
      ```

   b. At the SFTP prompt, enter this command to transfer the self-extracting archive file.

      ```
      put tkindbsrv-9.4_M2-n_lax.sh
      ```

7. (Optional) If your SFTP client does not copy the executable attribute from the client machine to the server, change the EXECUTE permission on the self-extracting archive file.

   ```
   chmod +x  tkindbsrv-9.4_M2-n_lax.sh
   ```

8. Unpack the self-extracting archive file in the partner directory.

   ```
   ./tkindbsrv-9.4_M2-n_lax.sh
   ```

   Note: You might need to add permissions for execution on this file. If so, do a **chmod +x** command on this file.

   This installs the SAS Embedded Process on the queen node. When Aster synchronizes the beehive, the files are copied to all the nodes. This can take a long time.

9. (Optional) There are two methods to copy the files to the nodes right away. You can do either of the following.

   - Run this code to manually move the files across all nodes on the beehive by using secure copy and SSH.

     ```
     location=/home/beehive/partner/
     cd $location
     for ip in `cat /home/beehive/cluster-management/hosts |
        grep node | awk '{print $3}'`; \
     ```

```
do \
echo $ip; \
scp -r SAS root@$ip":$location"; \
done
```

- Run this command to synchronize the beehive and restart the database.

  ```
  /home/beehive/bin/utils/primitives/UpgradeNCluster.py -u
  ```

10. Change to the directory where SAS is installed.

```
cd /home/beehive/partner/SAS/SASTKInDatabaseServerForAster/9.4_M2-2/sasexe
```

11. Install the SAS_SCORE( ), SAS_PUT( ), and other SQL/MR functions.

    a. Start the ACT tool.

    ```
    /home/beehive/clients/act -U db_superuser -w db_superuser-password
    -d database-to-install-sas_score-into
    ```

    b. (Optional) If this is not the first time you have installed the in-database deployment package for Aster, it is recommended that you remove the existing SQL/MR functions before installing the new ones. To do so, enter the following commands.

    ```
    \remove sas_score.tk.so
    \remove sas_put.tk.so
    \remove sas_row.tk.so
    \remove sas_partition.tk.so
    ```

    c. Enter the following commands to install the new SQL/MR functions. The SQL/MR functions need to be installed under the PUBLIC schema.

    ```
    \install sas_score.tk.so
    \install sas_put.tk.so
    \install sas_row.tk.so
    \install sas_partition.tk.so
    ```

12. Exit the ACT tool.

```
\q
```

13. Verify the existence and current date of the tkast-runInCluster and tkeastrmr.so files. These two binary files are needed by the SAS SQL/MR functions.

```
for ip in \
`cat /home/beehive/cluster-management/hosts | grep node | awk '{print $3}'`; \
   do \
   echo $ip; \
   ssh $ip "ls -al /home/beehive/partner/SAS/SASTKInDatabaseServerForAster/
      9.4_M2-2/sasexe/tkeastmr.so"; \
   ssh $ip "ls -al /home/beehive/partner/SAS/SASTKInDatabaseServerForAster/
      9.4_M2-2/utilities/bin/tkast-runInCluster"; \
done
```

Validating the Publishing of the SAS_SCORE( ) and the SAS_PUT( ) Functions

To validate that the SAS_SCORE( ) and the SAS_PUT( ) functions were installed, run the `\dF` command in the Aster Client or use any of the following views:

- **`nc_all_sqlmr_funcs`**, where **`all`** returns all functions on the system
- **`nc_user_sqlmr_funcs`**, where **`user`** returns all functions that are owned by or granted to the user
- **`nc_user_owned_sqlmr_funcs`**, where **`user_owned`** returns all functions that are owned by the user

Aster Permissions

The person who installs the in-database deployment package binary files in Aster needs root privileges for the queen node. This permission is most likely, but not necessarily, needed by the Aster system administrator.

For Aster 4.5, no permissions are needed by the person who runs the scoring or format publishing macros, because all functions and files are published to the PUBLIC schema.

For Aster 4.6 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
GRANT USAGE ON SCHEMA *yourschemaname* TO *youruserid*;

INSTALL FILE permission
GRANT INSTALL FILE ON SCHEMA *yourschemaname* TO *youruserid*;

CREATE permission
GRANT CREATE ON SCHEMA *yourschemaname* TO *youruserid*;

EXECUTE permission
GRANT EXECUTE ON FUNCTION PUBLIC.SAS_SCORE TO *youruserid*;

GRANT EXECUTE ON FUNCTION PUBLIC.SAS_PUT TO *youruserid*;

GRANT EXECUTE ON FUNCTION PUBLIC.SAS_ROW TO *youruserid*;

GRANT EXECUTE ON FUNCTION PUBLIC.SAS_PARTITION TO *youruserid*;

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*, located at http://support.sas.com/documentation/onlinedoc/indbtech/index.html

Chapter 3
Administrator’s Guide for DB2

In-Database Deployment Package for DB2

Prerequisites

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

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

Overview of the In-Database Deployment Package for DB2

This section describes how to install and configure the in-database deployment package for DB2 (SAS Formats Library for DB2 and SAS Embedded Process).

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 in-database deployment package for DB2 contains the SAS formats library and the precompiled binary files for two additional utility functions. The package also contains the SAS Embedded Process.

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 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.

The SAS Embedded Process is a SAS server process that runs within DB2 to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your DB2 system so that the SAS scoring files created in DB2 can access the routines within the SAS Embedded Process's run-time libraries.

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 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

1. If you are upgrading from or reinstalling a previous version, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 11.
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, and the SAS Embedded Process.

   For more information, see "Installing the SAS Formats Library, Binary Files, and SAS Embedded Process" on page 14.
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 20.
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 24.
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 11, "Configurations for SAS Model Manager," on page 109.

Upgrading from or Reinstalling a Previous Version

Overview of Upgrading from or Reinstalling a Previous Version

You can upgrade from or reinstall a previous version of the SAS Formats Library and binary files, the SAS Embedded Process, or both. See the following topics:

- If you want to upgrade or reinstall a previous version of the SAS Formats Library, binary files, and the SAS Embedded Process, see "Upgrading from or Reinstalling the SAS Formats Library, Binary Files, and the SAS Embedded Process" on page 12.
- If you want to upgrade or reinstall only the SAS Embedded Process, see "Upgrading from or Reinstalling the SAS Embedded Process" on page 13.

Upgrading from or Reinstalling the SAS Formats Library, Binary Files, and the SAS Embedded Process

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

Note: These steps also apply if you want to upgrade from or reinstall only the SAS Formats Library and binary files. If you want to upgrade from or reinstall only the SAS Embedded Process, see “Upgrading from or Reinstalling the SAS Embedded Process” on page 13.

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=/db2/9.4_M2/sqllib,
      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;
   ```

   If you are upgrading from or reinstalling only the SAS Formats Library and the binary files, skip to Step 6.

3. Enter the following command to see whether the SAS Embedded Process is running.

   ```
   $ps -ef | grep db2sasep
   ```

   If the SAS Embedded Process is running, results similar to this are displayed.

   ```
   ps -ef | grep db2sasep
   db2v9 23265382 20840668   0   Oct 06      -  4:03 db2sasep
   db2v9 27983990 16646196   1 08:24:09 pts/10  0:00 grep db2sasep
   ```

4. Stop the DB2 SAS Embedded Process using DB2IDA command.

   Use this command to stop the SAS Embedded Process.

   ```
   $db2ida -provider sas -stop
   ```

   If the SAS Embedded Process is still running, an error occurs. Enter this command to force the SAS Embedded Process to stop.

   ```
   $db2ida -provider sas -stopforce
   ```

   For more information about the DB2IDA command, see “Controlling the SAS Embedded Process for DB2” on page 19.

5. Remove the SAS directory that contain the SAS Embedded Process binary files from the DB2 instance path.

Enter these commands to move to the *db2instancepath* directory and remove the SAS directory. *db2instancepath* is the path to the SAS Embedded Process binary files in the DB2 instance.

```
$ cd db2instancepath
$ rm -fr SAS
```

6. 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
   ```

7. Remove the SAS directory from the DB2 instance path. Enter these commands to move to the *db2instancepath*/sqllib/function directory and remove the SAS directory. *db2instancepath*/sqllib/function is the path to the SAS_COMPILEUDF and SAS_DELETEUDF functions in the DB2 instance.

   ```
   $ cd db2instancepath/sqllib/function
   $ rm -fr SAS
   ```

Upgrading from or Reinstalling the SAS Embedded Process

To upgrade from or reinstall a previous version of the SAS Embedded Process, follow these steps.

Note: These steps are for upgrading from or reinstalling only the SAS Embedded Process. If you want to upgrade from or reinstall the SAS Formats Library and binary files or both the SAS Formats Library and binary files and the SAS Embedded Process, you must follow the steps in "Upgrading from or Reinstalling the SAS Formats Library, Binary Files, and the SAS Embedded Process" on page 12.

1. Enter the following command to see whether the SAS Embedded Process is running.

   ```
   $ps -ef | grep db2sasep
   ```

   If the SAS Embedded Process is running, results similar to this are displayed.

   ```
   ps -ef | grep db2sasep
   db2v9 23265382 20840668   0   Oct 06      -  4:03 db2sasep
   db2v9 27983990 16646196   1 08:24:09 pts/10  0:00 grep db2sasep
   ```

2. Enter the following command to determine whether there are any users connected to the instance.

   ```
   $db2 list applications
   ```

3. Stop the DB2 SAS Embedded Process using DB2IDA command.

   Note: If you are upgrading or reinstalling the SAS Embedded Process (tkindbsrv*.sh file), you do not need to shut down the database. The DB2IDA command enables you to upgrade or reinstall only the SAS Embedded Process components without impacting clients already connected to the database. For more information about the DB2IDA command, see "Controlling the SAS Embedded Process for DB2" on page 19.

Use this command to stop the SAS Embedded Process.

```
$db2ida -provider sas -stop
```

If the SAS Embedded Process is still running, an error occurs. Enter this command to force the SAS Embedded Process to stop.

```
$db2ida -provider sas -stopforce
```

4. Remove the SAS directory that contain the SAS Embedded Process binary files from the DB2 instance path.

   Enter these commands to move to the *db2instancepath* directory and remove the SAS directory. *db2instancepath* is the path to the SAS Embedded Process binary files in the DB2 instance.

   ```
   $ cd db2instancepath
   $ rm -fr SAS
   ```

Installing the SAS Formats Library, Binary Files, and SAS Embedded Process

Move the Files to DB2

There are two self-extracting archive files (.sh files) that need to be moved to DB2. You can use PSFTP, SFTP, or FTP to transfer the self-extracting archive files to the DB2 server to be unpacked and compiled.

- The first 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 ***SAS-installation-directory*/SASFormatsLibraryforDB2/3.1/DB2on<AIX | Linux64>/** directory.

  Choose the self-extracting archive files based on the UNIX platform that your DB2 server runs on. *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 reinstall or upgrade, *n* is incremented by 1.

  - AIX: acceldb2fmt-3.1-*n*_r64.sh
  - Linux(x86_64): acceldb2fmt-3.1-*n*_lax.sh

  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.

- The second self-extracting archive file contains the SAS Embedded Process. You need these files if you want to use the SAS Embedded Process to run your scoring model.

  Note: The SAS Embedded Process might require a later release of DB2 than function-based scoring. Please refer to the SAS system requirements documentation.

  This self-extracting archive file is located in the ***SAS-installation-directory*/SASTKInDatabaseServer/9.4/DB2on<AIX | Linuxx64>/**.

Choose the self-extracting archive files based on the UNIX platform that your DB2 server runs on. *n* is a number that indicates the latest version of the file.

- AIX: tkindbsrv-9.4_M2-*n*_r64.sh
- Linux(x86_64): tkindbsrv-9.4_M2-*n*_lax.sh

You must put the tkindbsrv file in the instance owner's home directory.

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

Unpack the SAS Formats Library and Binary Files

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 reinstall 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. If necessary, change permissions on the file to enable you to execute the script and write to the directory.

   ```
   $ chmod +x 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 are previous self-extracting archive files 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 /* rename SAS directory */
   $rm -fr SAS /* remove 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.

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
   ```

Unpack the SAS Embedded Process Files

After the tkindbsrv-9.4_M2-*n*_lax.sh or tkindbsrv-9.4_M2-*n*_r64.sh self-extracting archive file has been 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 reinstall 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 tkindbsrv file.

   ```
   $ cd path_to_sh_file
   ```

   path_to_sh_file is the location to which you copied the self-extracting archive file. This must be the instance owner home directory.

3. If necessary, change permissions on the file to enable you to execute the script and write to the directory.

   ```
   $ chmod +x tkindbsrv-9.4_M2-n_aix.sh
   ```

4. If there are previous self-extracting archive files 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 /* rename SAS directory */
   $rm -fr SAS /* remove SAS directory */
   ```

5. Use the following commands to unpack the appropriate self-extracting archive file.

   ```
   $ ./sh_file
   ```

sh_file is either tkindbsrv-9.4_M2-*n*_lax.sh or tkindbsrv-9.4_M2-*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 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.

```
/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.4_M2-n/bin
/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.4_M2-n/misc
/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.4_M2-n/sasexe
/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.4_M2-n/utilities
```

6. Use the DB2SET command to enable the SAS Embedded Process in DB2 and to tell the SAS Embedded Process where to find the SAS Embedded Process library files.

```
$ dbset DB2_SAS_SETTINGS="ENABLE_SAS_EP:true;
LIBRARY_PATH:db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.4_M2-n/sasexe"
```

   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 for SETENV commands. DB2SET registers the environment variable within DB2 only for the default database instance.

   For more information about all of the arguments that can be used with the DB2SET command for the SAS Embedded Process, see "DB2SET Command Syntax for the SAS Embedded Process" on page 18.

7. To verify that the SAS Embedded Process is set appropriately, run the DB2SET command without any parameters.

```
$ db2set
```

   The path should be similar to this one if it was set correctly. Note that the DB2LIBPATH that was set when you installed the SAS Formats Library and binary files is also listed.

```
DB2_SAS_SETTINGS=ENABLE_SAS_EP:true
   LIBRARY_PATH:db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.4_M2-n/sasexe
   DB2LIBPATH=db2instancepath/sqllib/function/SAS
```

8. Stop the database manager instance if it is not stopped already.

```
$ db2stop
```

   A message indicating that the stop was successful displays.

   If the database manager instance cannot be stopped because application programs are still connected to databases, use the FORCE APPLICATION command to disconnect all users, use the TERMINATE command to clear any background processes, and then use the DB2STOP command.

```
$ db2 list applications
$ db2 force applications all
$ db2 terminate
$ db2stop
```

9. (AIX only) Clear the cache.

```
$ su root
$ slibclean
$ exit
```

10. Restart the database manager instance.

```
$ db2start
```

11. Verify that the SAS Embedded Process started.

```
$ ps -ef | grep db2sasep
```

If the SAS Embedded Process was started, lines similar to the following are displayed.

```
ps -ef | grep db2sasep
db2v9 23265382 20840668   0   Oct 06      -  4:03 db2sasep
db2v9 27983990 16646196   1 08:24:09 pts/10  0:00 grep db2sasep
```

In the DB2 instance, you can also verify if the SAS Embedded Process log file was created in the DB2 instance's diagnostic directory.

```
$ cd  instance-home/sqllib/db2dump
$ ls -al sasep0.log
```

DB2SET Command Syntax for the SAS Embedded Process

The syntax for the DB2SET command 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*;>
<DIAGPATH:*path*;>
<DIAGLEVEL:*level-number*;>"

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 reinstall 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
```

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*/sqllib/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 23.

   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:

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

   For more information, see the "INDCONN Macro Variable" on page 21.

3. Run the %INDB2_PUBLISH_COMPILEUDF macro. For more information, see "%INDB2_PUBLISH_COMPILEUDF Macro Syntax" on page 23.

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 27.

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 24.

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*
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 11.

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 an error.

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 SAS_DELETEUDF functions.

%INDB2_PUBLISH_COMPILEUDF Macro Syntax

%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 the SAS Formats Library and Binary Files" on page 15.
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 21.

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 filename that the publishing macro uses to register the SAS_COMPILEUDF function. The object filename 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 filename.

OUTDIR=*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.

%INDB2_PUBLISH_DELETEUDF Macro Run Process

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 21.

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

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

   For more information, see the "INDCONN Macro Variable" on page 25.

3. Run the %INDB2_PUBLISH_DELETEUDF macro. For more information, see "%INDB2_PUBLISH_DELETEUDF Macro Syntax" on page 26.

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 27.

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 11.

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 24.

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_DELTEUDF 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/db2v9/sqllib
   ```

   In this example, **`/users/db2v9`** is the value of *db2instancepath* that was specified during installation and **`/users/db2v9/sqllib`** is also where the SAS_COMPILEUDF function was published.

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

   ```
   select funcname, implementation from syscat.functions where
      funcschema='SASLIB'
   ```

   You should receive a result similar to the following.

   ```
   FUNCNAME                         IMPLEMENTATION
   -------------------------------------------------------------
   SAS_DELETEUDF
   /users/db2v9/sqllib/function/SAS/SAS_DeleteUDF!SAS_DeleteUDF
   SAS_COMPILEUDF
   /users/db2v9/sqllib/function/SAS/SAS_CompileUDF!SAS_CompileUDF
   ```

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 SASUDF_COMPILER_PATH and SASUDF_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.

Permission Needed	Authority Required to Grant Permission	Examples
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		`GRANT CREATE_EXTERNAL_ROUTINE ON DATABASE TO` *`compiledeletepublisheruserid`*

- 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.

Permission Needed	Authority Required to Grant Permission	Examples
EXECUTE permission for functions that have been published. This enables the person who publishes the formats or scoring model functions to execute the SAS_COMPILEUDF and SAS_DELETEUDF functions.	System Administrator or Database Administrator *Note:* If you have SYSADM or DBADM authority, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions.	`GRANT EXECUTE ON FUNCTION SASLIB.* TO` *`scoringorfmtpublisherid`*
CREATE_EXTERNAL_ROUTINE permission to the database to create format or scoring model functions		`GRANT CREATE_EXTERNAL_ROUTINE ON DATABASE TO` *`scoringorfmtpublisherid`*
CREATE_NOT_FENCED_ROUTINE permission to create format or scoring model functions that are not fenced		`GRANT CREATE_NOT_FENCED_ROUTINE ON DATABASE TO` *`scoringorfmtpublisherid`*
CREATEIN permission for the schema in which the format or scoring model functions are published if the default schema (SASLIB) is not used		`GRANT CREATEIN ON SCHEMA` *`scoringschema`* `TO` *`scoringorfmtpublisherid`*
CREATE TABLE permission to create the model table used in with scoring and the SAS Embedded Process		`GRANT CREATETAB TO` *`scoringpublisherSEPid`*

READ permission to read the SASUDF_COMPILER_PATH and SASUDF_DB2PATH global variables
 	
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.
	Person who ran the %INDB2_PUBLISH_COMPILEUDF macro
	
	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.
		`GRANT READ ON VARIABLE SASLIB.SASUDF_DB2PATH TO` *`scoringorfmtpublisherid`*
		
		`GRANT READ ON VARIABLE SASLIB.SASUDF_COMPILER_PATH TO` *`scoringorfmtpublisherid`*

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 11, "Configurations for SAS Model Manager," on page 109.

Documentation for Using In-Database Processing in DB2

For information about how to publish SAS formats or scoring models, see the *SAS In-Database Products: User's Guide*, located at http://support.sas.com/documentation/onlinedoc/indbtech/index.html.

Chapter 4

Administrator's Guide to Greenplum

In-Database Deployment Package for Greenplum

Prerequisites

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

The SAS Scoring Accelerator for Greenplum requires 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.

Overview of the In-Database Deployment Package for Greenplum

This section describes how to install and configure the in-database deployment package for Greenplum (SAS Formats Library for Greenplum and the SAS Embedded Process).

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

  For more information, see the *SAS DS2 Language Reference*.
- 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, see the *SAS In-Database Products: User's Guide*.

The in-database deployment package for Greenplum contains the SAS formats library and precompiled binary files for the utility functions. The package also contains the SAS Embedded Process.

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 SAS Embedded Process is a SAS server process that runs within Greenplum to read and write data. The SAS Embedded Process contains the %INDGP_PUBLISH_COMPILEUDF_EP macro, run-time libraries, and other software that is installed on your Greenplum system. The %INDGP_PUBLISH_COMPILEUDF_EP macro defines the SAS_EP table function 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. 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 or reinstalling a previous release, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 33 before installing the in-database deployment package.
2. Install the SAS formats library, the binary files, and the SAS Embedded Process.

   For more information, see "Installing the SAS Formats Library, Binary Files, and SAS Embedded Process" on page 34.
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 11, "Configurations for SAS Model Manager," on page 109.

Note: If you are installing 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 or Reinstalling a Previous Version

Upgrading or Reinstalling the 9.3 SAS Formats Library and SAS Embedded Process

To upgrade from or reinstall the SAS 9.3 version, follow these steps:

1. Delete the ***`full-path-to-pkglibdir`*****`/SAS`** directory that contains the SAS Formats Library and the SAS Embedded Process.

   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 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 deleted. 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 or reinstall a previous version. Doing so ensures that you get the latest version of both the SAS Formats Library and the SAS Embedded Process.

2. Continue the installation instructions in "Installing the SAS Formats Library, Binary Files, and SAS Embedded Process" on page 34.

Upgrading or Reinstalling the 9.4 SAS Formats Library and SAS Embedded Process

To upgrade from or reinstall the SAS 9.4 version, follow these steps. If you upgrade or install the SAS Formats Library and the SAS Embedded Process 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.

   ```
   ./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.

   The UninstallSASEPFiles.sh file is in the **`path_to_sh_file`** directory where you copied the tkindbsrv-9.4_M2-*n*_lax.sh self-extracting archive file.

   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.

   ```
   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, Binary Files, and SAS Embedded Process" on page 34.

Installing the SAS Formats Library, Binary Files, and SAS Embedded Process

Moving and 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 ***SAS-***

***installation-directory*/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 reinstall 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 **`\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:

   ```
   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.

   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.

Moving and Installing the SAS Embedded Process

The SAS Embedded Process is contained in a self-extracting archive file. The self-extracting archive file is located in the ***SAS-installation-directory*/SASTKInDatabaseServer/9.4/GreenplumonLinux64** directory.

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

1. Using a method of your choice, transfer the tkindbsrv-9.4_M2-*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 reinstall 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 at a later time.

2. After the tkindbsrv-9.4_M2-*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.

   ```
   ./tkindbsrv-9.4_M2-n_lax.sh
   ```

   Note: If you receive a "permissions denied" message, check the permissions on the tkindbsrv-9.4_M2-*n*_lax.sh file. This file must have EXECUTE permissions to run.

   After the script runs and the files are unpacked, the contents of the target directories should look similar to these. *path_to_sh_file* is the location to which you copied the self-extracting archive file in Step 1.

   ```
   /path_to_sh_file/InstallSASEPFiles.sh
   /path_to_sh_file/UninstallSASEPFiles.sh
   /path_to_sh_file/StartupSASEP.sh
   /path_to_sh_file/ShutdownSASEP.sh
   /path_to_sh_file/ShowSASEPStatus.sh
   /path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/admin
   /path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/bin
   /path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/logs
   /path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/misc
   /path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/sasexe
   /path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/utilities
   ```

   Note: In addition to the ***/path_to_sh_file/*** directory, all of the .sh files are also placed in the ***/path_to_sh_file/*SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/admin** directory.

   The InstallSASEPFiles.sh file installs the SAS Embedded Process. The next step explains how to run this file. The StartupSASEP.sh and ShutdownSASEP.sh files enable you to manually start and stop the SAS Embedded Process. For more information about running these two files, see "Controlling the SAS Embedded Process" on page 46.

The UninstallSASEPFiles.sh file uninstalls the SAS Embedded Process. The ShowEPFilesStatus.sh file shows the status of the SAS Embedded Process on each host.

CAUTION:

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

5. Use the following commands at the UNIX prompt to install the SAS Embedded Process on the master node.

   The InstallSASEPFiles.sh file must be run from the */path_to_sh_file/* directory.

   ```
   cd /path_to_sh_file/
   ./InstallSASEPFiles.sh <-quiet>
   ```

   Note: *-verbose* is on by default and enables you to see all messages generated during the installation process. Specify *-quiet* to suppress messages.

   The installation deploys the SAS Embedded Process to all the host nodes automatically.

   The installation also creates a ***`full-path-to-pkglibdir`*`/SAS`** directory. This directory is created on the master node and each host node.

   The installation also copies the SAS directories and files from Step 4 across every node.

   The contents of the ***`full-path-to-pkglibdir/`*** **`SAS/SASTKInDatabaseServerForGreenplum`** directory should look similar to these.

   ```
   full-path-to-pkglibdir/SAS/SASTKInDatabaseServerForGreenplum/
      9.4_M2/admin
   full-path-to-pkglibdir/SAS/SASTKInDatabaseServerForGreenplum/
      9.4_M2/bin
   full-path-to-pkglibdir/SAS/SASTKInDatabaseServerForGreenplum/
      9.4_M2/logs
   full-path-to-pkglibdir/SAS/SASTKInDatabaseServerForGreenplum/
      9.4_M2/misc
   full-path-to-pkglibdir/SAS/SASTKInDatabaseServerForGreenplum/
      9.4_M2/sasexe
   full-path-to-pkglibdir/SAS/SASTKInDatabaseServerForGreenplum/
      9.4_M2/utilities
   ```

   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.

   This is an example of a SAS directory.

   ```
   usr/local/greenplum-db-4.2.3.0/lib/postgresql/SAS
   ```

Running the %INDGP_PUBLISH_COMPILEUDF Macro

Overview of the %INDGP_PUBLISH_COMPILEUDF Macro

Use the %INDGP_PUBLISH_COMPILEUDF macro if you want to 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 that are used when the scoring publishing macro transfers 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 39.

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*<'>
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*<'>
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*<'>
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*<'>
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*<'>
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*<'>
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 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 from or reinstalling 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 or Reinstalling a Previous Version" on page 33 and "Moving and Installing the SAS Formats Library and Binary Files" on page 34.

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=*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 function 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*<'>
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*<'>
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*<'>
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*<'>
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*<'>
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*<'>
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*<'>
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 from or reinstalling 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 or Reinstalling a Previous Version" on page 33 and "Moving and Installing the SAS Embedded Process" on page 36.

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=*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.

   ```
   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.

   ```
   psql -d databasename
   ```

   You should receive the following prompt.

   ```
   databasename=#
   ```

2. At the prompt, enter the following command.

   ```
   select prosrc, probin from pg_catalog.pg_proc where proname = 'sas_ep';
   ```

   You should receive a result similar to the following:

   ```
   SAS_EP | $libdir/SAS/sasep_tablefunc.so
   ```

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 “Moving and Installing the SAS Embedded Process” on page 36.

```
/path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2
```

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

```
/path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/ShutdownSASEP.sh
    <-quiet>
```

When invoked from the master node, ShutdownSASEP.sh shuts down the SAS Embedded Process on each database node. 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. 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.

```
/path_to_sh_file/SAS/SASTKInDatabaseServerForGreenplum/9.4_M2/StartupSASEP.sh
    <-quiet>
```

When invoked from the master node, StartupSASEP.sh manually starts the SAS Embedded Process on each database node. The *-verbose* option is on by default and provides a status of the installation as it occurs. You can specify the *-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.

CAUTION:

The timing option must be off for any of the .sh scripts to work. Put **`\timing off`** in your .psqlrc file before running these scripts.

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:

```
maximum_number_semaphore_arrays = 5 * 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 11, “Configurations for SAS Model Manager,” on page 109.

Documentation for Using In-Database Processing in Greenplum

For information about how to publish SAS formats and scoring models, see the *SAS In-Database Products: User's Guide*, located at http://support.sas.com/documentation/onlinedoc/indbtech/index.html.

For information about how to use the SAS In-Database Code Accelerator, see the *SAS DS2 Language Reference*, located at http://support.sas.com/documentation/onlinedoc/base/index.html.

Chapter 5
Administrator's Guide for Hadoop

In-Database Deployment Package for Hadoop

Prerequisites

The following prerequisites are required before you install and configure the in-database deployment package for Hadoop:

- SAS Foundation and the SAS/ACCESS Interface to Hadoop are installed.
- You have working knowledge of the Hadoop vendor distribution that you are using (for example, Cloudera or Hortonworks).

  You also need working knowledge of the Hadoop Distributed File System (HDFS), MapReduce 1, MapReduce 2, YARN, Hive, and HiveServer2 services. For more information, see the Apache website or the vendor's website.
- The SAS Scoring Accelerator for Hadoop requires a specific configuration file. For more information, see "How to Merge Configuration File Properties" on page 58.

- The HDFS, MapReduce, YARN, and Hive services must be running on the Hadoop cluster.
- The SAS Scoring Accelerator for Hadoop requires a specific version of the Hadoop distribution. For more information, see the SAS Foundation system requirements documentation for your operating environment.
- You have root or sudo access. Your user has Write permission to the root of HDFS.
- You know the location of the MapReduce home.
- You know the host name of the Hive server and the NameNode.
- You understand and can verify your Hadoop user authentication.
- You understand and can verify your security setup.

  If you are using Kerberos, you need the ability to get a Kerberos ticket.
- You have permission to restart the Hadoop MapReduce service.
- In order to avoid SSH key mismatches during installation, add the following two options to the SSH **config** file, under the user's home .ssh folder. An example of a home .ssh folder is **/root/.ssh/**. *nodes* is a list of nodes separated by a space.

  ```
  host nodes
     StrictHostKeyChecking no
     UserKnownHostsFile /dev/null
  ```

  For more details about the SSH **config** file, see the SSH documentation.
- All machines in the cluster are set up to communicate with passwordless SSH. Verify that the nodes can access the node that you chose to be the master node by using SSH.

  Traditionally, public key authentication in Secure Shell (SSH) is used to meet the passwordless access requirement. SSH keys can be generated with the following example.

  ```
  [root@raincloud1 .ssh]# ssh-keygen -t rsa
  Generating public/private rsa key pair.
  Enter file in which to save the key (/root/.ssh/id_rsa):
  Enter passphrase (empty for no passphrase):
  Enter same passphrase again:
  Your identification has been saved in /root/.ssh/id_rsa.
  Your public key has been saved in /root/.ssh/id_rsa.pub.
  The key fingerprint is:
  09:f3:d7:15:57:8a:dd:9c:df:e5:e8:1d:e7:ab:67:86 root@raincloud1

  add id_rsa.pub public key from each node to the master node authorized
     key file under /root/.ssh/authorized_keys
  ```

  For Secure Mode Hadoop, GSSAPI with Kerberos is used as the passwordless SSH mechanism. GSSAPI with Kerberos not only meets the passwordless SSH requirements, but also supplies Hadoop with the credentials required for users to perform operations in HDFS with SAS LASR Analytic Server and SASHDAT files. Certain options must be set in the SSH daemon and SSH client configuration files. Those options are as follows and assume a default configuration of sshd.

  To configure passwordless SSH to use Kerberos, follow these steps:

  1. In the sshd_config file, set:

  ```
  GSSAPIAuthentication yes
  ```

2. In the ssh_config file, set:

```
Host *.domain.net
GSSAPIAuthentication yes
GSSAPIDelegateCredentials yes
```

where *domain.net* is the domain name used by the machine in the cluster.

TIP Although you can specify **`host *`**, this is not recommended because it allows GSSAPI Authentication with any host name.

Overview of the In-Database Deployment Package for Hadoop

This section describes how to install and configure the in-database deployment package for Hadoop (SAS Embedded Process).

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

- Run a scoring model in Hadoop Distributed File System (HDFS) using the SAS Scoring Accelerator for Hadoop.
- Run DATA step scoring programs in Hadoop.
- Run DS2 threaded programs in Hadoop using the SAS In-Database Code Accelerator for Hadoop.
- 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*.
- Transform data in Hadoop and extract transformed data out of Hadoop for analysis in SAS with the SAS Data Loader for Hadoop. For more information, see *SAS Data Loader for Hadoop: User's Guide*.
- Perform data quality operations in Hadoop using the SAS Data Loader for Hadoop. For more information, see *SAS Data Loader for Hadoop: User's Guide*.

The in-database deployment package for Hadoop includes the SAS Embedded Process and two SAS Hadoop MapReduce JAR files. The SAS Embedded Process is a SAS server process that runs within Hadoop to read and write data. The SAS Embedded Process contains macros, run-time libraries, and other software that is installed on your Hadoop system.

The SAS Embedded Process must be installed on all nodes capable of executing either MapReduce 1 or MapReduce 2 tasks. For MapReduce 1, this would be nodes where a TaskTracker is running. For MapReduce 2, this would be nodes where a NodeManager is running. Usually, every DataNode node has a YARN NodeManager or a MapReduce 1 TaskTracker running. By default, the SAS Embedded Process install script (sasep-servers.sh) discovers the cluster topology and installs the SAS Embedded Process on all DataNode nodes, including the host node from where you run the script (the Hadoop master NameNode). This occurs even if a DataNode is not present. If you want to limit the list of nodes on which you want the SAS Embedded Process installed, you should run the sasep-servers.sh script with the **`-host`** <*hosts*> option. The SAS Hadoop MapReduce JAR files must be installed on all nodes of a Hadoop cluster.

Hadoop Installation and Configuration

Hadoop Installation and Configuration Steps

Before you begin the Hadoop installation and configuration, review "Prerequisites" on page 49.

1. If you are upgrading from or reinstalling a previous release, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 52 before installing the in-database deployment package.
2. Move the SAS Embedded Process and SAS Hadoop MapReduce JAR file install scripts to the Hadoop master node (the NameNode).

   For more information, see "Moving the SAS Embedded Process and SAS Hadoop MapReduce JAR File Install Scripts" on page 54.

   Note: The location where you transfer the install scripts becomes the SAS Embedded Process home and is referred to as *SASEPHome* throughout this chapter.

   Note: Both the SAS Embedded Process install script and the SAS Hadoop MapReduce JAR file install script must be transferred to the *SASEPHome* directory.
3. Install the SAS Embedded Process and the SAS Hadoop MapReduce JAR files.

   For more information, see "Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files" on page 55.
4. If you want to use the SAS Scoring Accelerator for Hadoop, merge the properties of several configuration files into one configuration file.

   For more information, see "How to Merge Configuration File Properties" on page 58.
5. Copy the Hadoop core and common Hadoop JAR files to the client machine.

   For more information, see "Copying Hadoop JAR Files to the Client Machine" on page 60.

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*.

Upgrading from or Reinstalling a Previous Version

To upgrade or reinstall a previous version, follow these steps.

1. If you are upgrading from SAS 9.3, follow these steps. If you are upgrading from SAS 9.4, start with Step 2.

   a. Stop the Hadoop SAS Embedded Process.

   ```
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.35/bin/sasep-stop.all.sh
   ```

   SASEPHome is the master node where you installed the SAS Embedded Process.

b. Delete the Hadoop SAS Embedded Process from all nodes.

```
SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.35/bin/sasep-delete.all.sh
```

c. Verify that the sas.hadoop.ep.*distribution-name*.jar files have been deleted.

The JAR files are located at ***HadoopHome*/lib**.

For Cloudera, the JAR files are typically located here:

```
/opt/cloudera/parcels/CDH/lib/hadoop/lib
```

For Hortonworks, the JAR files are typically located here:

```
/usr/lib/hadoop/lib
```

d. Continue with Step 3.

2. If you are upgrading from SAS 9.4, follow these steps.

a. Stop the Hadoop SAS Embedded Process.

```
SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.*/bin/sasep-servers.sh
   -stop -hostfile host-list-filename | -host <">host-list<">
```

SASEPHome is the master node where you installed the SAS Embedded Process.

For more information, see "SASEP-SERVERS.SH Script" on page 61.

b. Remove the SAS Embedded Process from all nodes.

```
SASEPHome/SAS/SASTKInDatabaseForServerHadoop/9.*/bin/sasep-servers.sh
   -remove -hostfile host-list-filename | -host <">host-list<">
   -mrhome dir
```

Note: This step ensures that all old SAS Hadoop MapReduce JAR files are removed.

For more information, see "SASEP-SERVERS.SH Script" on page 61.

c. Verify that the sas.hadoop.ep.apache*.jar files have been deleted.

The JAR files are located at ***HadoopHome*/lib**.

For Cloudera, the JAR files are typically located here:

```
/opt/cloudera/parcels/CDH/lib/hadoop/lib
```

For Hortonworks, the JAR files are typically located here:

```
/usr/lib/hadoop/lib
```

d. Manually remove the SAS Embedded Process directories and files on the node from which you ran the script.

The sasep-servers.sh -remove script removes the file everywhere except on the node from which you ran the script. The sasep-servers.sh -remove script displays instructions that are similar to the following example.

```
localhost WARN: Apparently, you are trying to uninstall SAS Embedded Process
   for Hadoop from the local node.
The binary files located at
   local_node/SAS/SASTKInDatabaseServerForHadoop/local_node/
SAS/SASACCESStoHadoopMapReduceJARFiles will not be removed.
localhost WARN: The init script will be removed from /etc/init.d and the
   SAS Map Reduce JAR files will be removed from /usr/lib/hadoop-mapreduce/lib.
localhost WARN: The binary files located at local_node/SAS
   should be removed manually.
```

3. Continue the installation process.

   For more information, see "Moving the SAS Embedded Process and SAS Hadoop MapReduce JAR File Install Scripts" on page 54.

Moving the SAS Embedded Process and SAS Hadoop MapReduce JAR File Install Scripts

Creating the SAS Embedded Process Directory

Before you can install the SAS Embedded Process and the SAS Hadoop MapReduce JAR files, you must move the SAS Embedded Process and SAS Hadoop MapReduce JAR file install scripts to a directory on the Hadoop master node (the NameNode).

Create a new directory that is not part of an existing directory structure, such as **`/sasep`**.

This path will be created on each node in the Hadoop cluster during the SAS Embedded Process installation. Do not use existing system directories such as **`/opt`** or **`/usr`**. This new directory becomes the SAS Embedded Process home and is referred to as *SASEPHome* throughout this chapter.

Moving the SAS Embedded Process Install Script

The SAS Embedded Process install script is contained in a self-extracting archive file named tkindbsrv-9.42-*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 reinstall or upgrade, *n* is incremented by 1. The self-extracting archive file is located in the ***`SAS-installation-directory`*****`/SASTKInDatabaseServer/9.4/HadooponLinuxx64/`** directory.

Using a method of your choice, transfer the SAS Embedded Process install script to your Hadoop master node.

This example uses secure copy, and *SASEPHome* is the location where you want to install the SAS Embedded Process.

```
scp tkindbsrv-9.42-n_lax.sh username@hadoop:/SASEPHome
```

Note: Both the SAS Embedded Process install script and the SAS Hadoop MapReduce JAR file install script must be transferred to the *SASEPHome* directory.

Moving the SAS Hadoop MapReduce JAR File Install Script

The SAS Hadoop MapReduce JAR file install script is contained in a self-extracting archive file named hadoopmrjars-9.42-*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 reinstall or upgrade, *n* is incremented by 1. The self-extracting archive file is located in the ***`SAS-installation-directory/`*****`SASACCESStoHadoopMapReduceJARFiles/9.41`** directory.

Using a method of your choice, transfer the SAS Hadoop MapReduce JAR file install script to your Hadoop master node.

This example uses Secure Copy, and *SASEPHome* is the location where you want to install the SAS Hadoop MapReduce JAR files.

```
scp hadoopmrjars-9.42-n_lax.sh username@hadoop:/SASEPHome
```

Note: Both the SAS Embedded Process install script and the SAS Hadoop MapReduce JAR file install script must be transferred to the *SASEPHome* directory.

Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files

To install the SAS Embedded Process, follow these steps.

Note: Permissions are needed to install the SAS Embedded Process and SAS Hadoop MapReduce JAR files. For more information, see “Hadoop Permissions” on page 67.

1. Log on to the server using SSH as root with sudo access.

   ```
   ssh username@serverhostname
   sudo su - root
   ```

2. Move to your Hadoop master node where you want the SAS Embedded Process installed.

   ```
   cd /SASEPHome
   ```

   SASEPHome is the same location to which you copied the self-extracting archive file. For more information, see “Moving the SAS Embedded Process Install Script” on page 54.

   Note: Before continuing with the next step, ensure that each self-extracting archive file has Execute permission.

3. Use the following script to unpack the tkindbsrv-9.42-*n*_lax.sh file.

   ```
   ./tkindbsrv-9.42-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 reinstall or upgrade, *n* is incremented by 1.

   Note: If you unpack in the wrong directory, you can move it after the unpack.

   After this script is run and the files are unpacked, the script creates the following directory structure where *SASEPHome* is the master node from Step 1.

   ```
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/misc
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/sasexe
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/utilities
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/build
   ```

   The content of the **`SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin`** directory should look similar to this.

   ```
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sas.ep4hadoop.template
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sasep-servers.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sasep-common.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sasep-server-start.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sasep-server-status.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sasep-server-stop.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/InstallTKIndbsrv.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/MANIFEST.MF
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/qkbpush.sh
   SASEPHome/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/sas.tools.qkb.hadoop.jar
   ```

4. Use this command to unpack the SAS Hadoop MapReduce JAR files.

   ```
   ./hadoopmrjars-9.42-n_lax.sh
   ```

After the script is run, the script creates the following directory and unpacks these files to that directory.

```
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/ep-config.xml
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/
   sas.hadoop.ep.apache023.jar
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/
    sas.hadoop.ep.apache023.nls.jar
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/
   sas.hadoop.ep.apache121.jar
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/
   sas.hadoop.ep.apache121.nls.jar
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/
   sas.hadoop.ep.apache205.jar
SASEPHome/SAS/SASACCESStoHadoopMapReduceJARFiles/9.42/lib/
   sas.hadoop.ep.apache205.nls.jar
```

5. Use the **`sasep-servers.sh -add`** script to deploy the SAS Embedded Process installation across all nodes. The SAS Embedded Process is installed as a Linux service.

   TIP There are many options available when installing the SAS Embedded Process. We recommend that you review the script syntax before running it. For more information, see "SASEP-SERVERS.SH Script" on page 61.

   Note: If you are running on a cluster with Kerberos, complete both steps a and b. If you are not running with Kerberos, complete only step b.

   a. If you are running on a cluster with Kerberos, you must kinit the HDFS user.

      ```
      sudo su - root
      su - hdfs | hdfs-userid
      kinit -kt location of keytab file
         user for which you are requesting a ticket
      exit
      ```

      Here is an example:

      ```
      sudo su - root
      su - hdfs
      kinit -kt hdfs.keytab hdfs
      exit
      ```

      Note: The default HDFS user is **`hdfs`**. You can specify a different user ID with the -hdfsuser argument when you run the **`sasep-servers.sh -add`** script.

      Note: If you are running on a cluster with Kerberos, a keytab is required for the -hdfsuser running the **`sasep-servers.sh -add`** script.

      Note: You can run klist while you are running as the -hdfsuser user to check the status of your Kerberos ticket on the server. Here is an example:

```
klist
Ticket cache: FILE/tmp/krb5cc_493
Default principal: hdfs@HOST.COMPANY.COM

Valid starting    Expires           Service principal
06/20/14 09:51:26 06/27/14 09:51:26 krbtgt/HOST.COMPANY.COM@HOST.COMPANY.COM
     renew until 06/22/14 09:51:26
```

b. Run the sasep-servers.sh script. Review all of the information in this step before running the script.

```
cd SASEPHOME/SAS/SASTKInDatabaseServerForHadoop/9.42/bin
./sasep-servers.sh -add
```

During the install process, the script asks whether you want to start the SAS Embedded Process. If you choose **`Y`** or **`y`**, the SAS Embedded Process is started on all nodes after the install is complete. If you choose **`N`** or **`n`**, you can start the SAS Embedded Process later by running **`./sasep-servers.sh -start`**.

Note: When you run the **`sasep-servers.sh -add`** script, a user and group named **`sasep`** is created. You can specify a different user and group name with the -epuser and -epgroup arguments when you run the **`sasep-servers.sh -add`** script.

Note: The sasep-servers.sh script can be run from any location. You can also add its location to the PATH environment variable.

TIP Although you can install the SAS Embedded Process in multiple locations, the best practice is to install only one instance. Only one version of the SASEP JAR files will be installed in your *HadoopHome*/lib directory.

Note: The SAS Embedded Process must be installed on all nodes capable of executing either MapReduce 1 or MapReduce 2 tasks. For MapReduce 1, this would be nodes where a TaskTracker is running. For MapReduce 2, this would be nodes where a NodeManager is running. Usually, every DataNode node has a YARN NodeManager or a MapReduce 1 TaskTracker running. By default, the SAS Embedded Process install script (sasep-servers.sh) discovers the cluster topology and installs the SAS Embedded Process on all DataNode nodes, including the host node from where you run the script (the Hadoop master NameNode). This occurs even if a DataNode is not present. If you want to limit the list of nodes on which you want the SAS Embedded Process installed, you should run the sasep-servers.sh script with the **`-host`** <*hosts*> option.

Note: If you install the SAS Embedded Process on a large cluster, the SSHD daemon might reach the maximum number of concurrent connections. The `ssh_exchange_identification: Connection closed by remote host` SSHD error might occur. To work around the problem, edit the /etc/ssh/sshd_config file, change the *MaxStartups* option to the number that accommodates your cluster, and save the file. Then, reload the SSHD daemon by running the /etc/init.d/sshd reload command.

6. Verify that the SAS Embedded Process is installed and running. Change directories and then run the sasep-servers.sh script with the **`-status`** option.

```
cd SASEPHOME/SAS/SASTKInDatabaseServerForHadoop/9.42/bin
./sasep-servers.sh -status
```

This command returns the status of the SAS Embedded Process running on each node of the Hadoop cluster. Verify that the SAS Embedded Process home directory is correct on all the nodes.

Note: The sasep-servers.sh -status script will not run successfully if the SAS Embedded Process is not installed.

7. Verify that the sas.hadoop.ep.apache*.jar files are now in place on all nodes.

The JAR files are located at ***`HadoopHome`*`/lib`**.

For Cloudera, the JAR files are typically located here:

```
/opt/cloudera/parcels/CDH/lib/hadoop/lib
```

For Hortonworks, the JAR files are typically located here:

```
/usr/lib/hadoop/lib
```

8. Restart the Hadoop YARN or MapReduce service.

   This enables the cluster to reload the SAS Hadoop JAR files (sas.hadoop.ep.*.jar).

   Note: It is preferable to restart the service by using Cloudera Manager or Hortonworks Ambari.

9. Verify that an init.d service with a sas.ep4hadoop file was created in the following directory.

   ```
   /etc/init.d
   ```

   View the sas.ep4hadoop file and verify that the SAS Embedded Process home directory is correct.

   The init.d service is configured to start at level 3 and level 5.

   Note: The SAS Embedded Process needs to run on all nodes in the Hadoop cluster.

10. Verify that the configuration file, ep-config.xml, was written to the HDFS file system.

    ```
    hadoop fs -ls /sas/ep/config
    ```

    Note: If you are running on a cluster with Kerberos, you need a Kerberos ticket. If not, you can use the WebHDFS browser.

    Note: The **`/sas/ep/config`** directory is created automatically when you run the install script.

How to Merge Configuration File Properties

Requirements for Configuration File Properties

The SAS Scoring Accelerator for Hadoop requires that some specific configuration files be merged into a single configuration file. This configuration file is used by the %INDHD_RUN_MODEL macro. For more information about the %INDHD_RUN_MODEL macro, see the *SAS In-Database Products: User's Guide*.

Note: In addition to the SAS Scoring Accelerator for Hadoop, a merged configuration file is also required for using PROC HADOOP and the FILENAME Statement Hadoop Access Method with Base SAS.

Note: A merged configuration file is not required for the following products:

- SAS/ACCESS Interface to Hadoop
- Scalable Performance Data Engine (SPD Engine)
- High-Performance Analytics
- SAS Code Accelerator for Hadoop

In the August 2014 release, a new environment variable, SAS_HADOOP_CONFIG_PATH, was created to replace the use of a merged configuration file for the preceding list of products. The configuration files are copied to a location on the client machine, and the SAS_HADOOP_CONFIG_PATH variable is set to that location.

Note: The configuration file properties that must be merged depend on which version of Hadoop you are using. The following topics are for Cloudera CDH4.*x* and 5.x and

Hortonworks 1.3.2 and 2.*x*. If you are using a different version of Cloudera or Hortonworks or another vendor's Hadoop distribution, you must use a comparable version of these configuration files. Otherwise, the installation of the SAS Embedded Process will fail.

How to Merge Configuration File Properties

To merge configuration file properties, follow these steps:

1. Retrieve the Hadoop configuration files from this location on your cluster.

   `/etc/hadoop/conf`

2. Using a method of your choice, concatenate the required configuration files into one configuration file.

   a. If you are using MapReduce 1, merge the properties from the Hadoop core (core-site.xml), Hadoop HDFS (hdfs-site.xml), and MapReduce (mapred-site.xml) configuration files into one single configuration file.

   b. If you are using MapReduce 2 or YARN, merge the properties from the Hadoop core (core-site.xml), Hadoop HDFS (hdfs-site.xml), MapReduce (mapred-site.xml), and YARN (yarn-site.xml) configuration files into one single configuration file.

   Note: The merged configuration file must have one beginning <configuration> tag and one ending </configuration> tag. Only properties should exist between the <configuration>…</configuration> tags. Here is a Cloudera example:

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

<configuration>
  <property>
    <name>hive.metastore.local</name>
    <value>false</value>
  </property>

<!-- lines omitted for sake of brevity -->

  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value></value>
  </property>
</configuration>
```

   Note: For information about setting a configuration property to avoid out of memory exceptions, see "Configuring the Number of Proactive Reads in the SAS Embedded Process" on page 59.

3. Save the configuration file to a location of your choosing.

Configuring the Number of Proactive Reads in the SAS Embedded Process

The SAS Embedded Process for Hadoop implements a mechanism that proactively reads and caches records from the input files while the DS2 program is processing the current block of records. By default, the number of proactive reads is set to 100. On files with very large records, the default value of 100 might cause memory exhaustion in the Java Virtual Machine in which the SAS Embedded Process MapReduce task is running.

TIP Java Virtual Machine out of memory exceptions might be avoided if the number of proactive reads is reduced.

To avoid out of memory exceptions, we recommend setting the following property in the merged configuration file:

```
<property>
   <name>sas.ep.superreader.proactive.reader.capacity</name>
   <value>10</value>
</property>
```

Copying Hadoop JAR Files to the Client Machine

For SAS components that interface with Hadoop, a specific set of common and core Hadoop JAR files must be in one location on the client machine. Examples of those components are the SAS Scoring Accelerator and SAS High-Performance Analytics.

When you run the **sasep-servers.sh -add** script to install the SAS Embedded Process, the script detects the Hadoop distribution and creates a HADOOP_JARS.zip file in the ***SASEPHome*/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/** directory. This file contains the common and core Hadoop JAR files that are required for the SAS Embedded Process. For more information, see "Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files" on page 55.

To get the Hadoop JAR files on your client machine, follow these steps:

1. Move the HADOOP_JARS.zip file to a directory on your client machine and unzip the file.

   ```
   unzip HADOOP_JARS.zip
   ```

2. Set the SAS_HADOOP_JAR_PATH environment variable to point to the directory that contains the core and common Hadoop JAR files.

Note: You can run the **sasep-servers.sh -getjars** script at any time to create a new ZIP file and refresh the JAR file list.

Note: The MapReduce 1 and MapReduce 2 JAR files cannot be on the same Java classpath.

Note: The JAR files in the SAS_HADOOP_JAR_PATH directory must match the Hadoop server to which SAS is connected. If multiple Hadoop servers are running different Hadoop versions, then create and populate separate directories with version-specific Hadoop JAR files for each Hadoop version. Then dynamically set SAS_HADOOP_JAR_PATH, based on the target Hadoop server to which each SAS job or SAS session is connected. One way to dynamically set SAS_HADOOP_JAR_PATH is to create a wrapper script associated with each Hadoop version. Then invoke SAS via a wrapper script that sets SAS_HADOOP_JAR_PATH appropriately to pick up the JAR files that match the target Hadoop server. Upgrading your Hadoop server version might involve multiple active Hadoop versions. The same multi-version instructions apply.

SASEP-SERVERS.SH Script

Overview of the SASEP-SERVERS.SH Script

The sasep-servers.sh script enables you to perform the following actions.

- Install or uninstall the SAS Embedded Process and SAS Hadoop MapReduce JAR files on a single node or a group of nodes.
- Start or stop the SAS Embedded Process on a single node or on a group of nodes.
- Determine the status of the SAS Embedded Process on a single node or on a group of nodes.
- Write the installation output to a log file.
- Pass options to the SAS Embedded Process.
- Create a HADOOP_JARS.zip file in the local folder. This ZIP file contains all required client JAR files.

Note: The sasep-servers.sh script can be run from any folder on any node in the cluster. You can also add its location to the PATH environment variable.

Note: You must have sudo access to run the sasep-servers.sh script.

SASEP-SERVERS.SH Syntax

sasep-servers.sh

-add | -remove | -start | -stop | -status | -restart

<-mrhome *path-to-mr-home*>

<-hdfsuser *user-id*>

<-epuser>*epuser-id*

<-epgroup>*epgroup-id*

<-hostfile *host-list-filename* | -host <">*host-list*<">>

<-epscript *path-to-ep-install-script*>

<-mrscript *path-to-mr-jar-file-script*>

<-options "*option-list*">

<-log *filename*>

<-version *apache-version-number*>

<-getjars>

Arguments

-add

installs the SAS Embedded Process.

Note The -add argument also starts the SAS Embedded Process (same function as -start argument). You are prompted and can choose whether to start the SAS Embedded Process.

Tip You can specify the hosts on which you want to install the SAS Embedded Process by using the -hostfile or -host option. The -hostfile or -host options are mutually exclusive.

See -hostfile and -host option on page 63

-remove
removes the SAS Embedded Process.

CAUTION:
If you are using SAS Data Loader's Cleanse Data in Hadoop directives, you should remove the QKB from the Hadoop nodes before removing the SAS Embedded Process. The QKB is removed by running the QKBPUSH script. For more information, see the *SAS Data Loader for Hadoop: Administrator's Guide*.

Tip You can specify the hosts for which you want to remove the SAS Embedded Process by using the -hostfile or -host option. The -hostfile or -host options are mutually exclusive.

See -hostfile and -host option on page 63

-start
starts the SAS Embedded Process.

Tip You can specify the hosts on which you want to start the SAS Embedded Process by using the -hostfile or -host option. The -hostfile or -host options are mutually exclusive.

See -hostfile and -host option on page 63

-stop
stops the SAS Embedded Process.

Tip You can specify the hosts on which you want to stop the SAS Embedded Process by using the -hostfile or -host option. The -hostfile or -host options are mutually exclusive.

See -hostfile and -host option on page 63

-status
provides the status of the SAS Embedded Process on all hosts or the hosts that you specify with either the -hostfile or -host option.

Tips The status also shows the version and path information for the SAS Embedded Process.

You can specify the hosts for which you want the status of the SAS Embedded Process by using the -hostfile or -host option. The -hostfile or -host options are mutually exclusive.

See -hostfile and -host option on page 63

-restart
restarts the SAS Embedded Process.

Tip	You can specify the hosts on which you want to restart the SAS Embedded Process by using the -hostfile or -host option. The -hostfile or -host options are mutually exclusive.
See	-hostfile and -host option on page 63

-mrhome ***path-to-mr-home***
specifies the path to the MapReduce home.

-hdfsuser ***user-id***
specifies the user ID that has Write access to HDFS root directory.

Default	hdfs
Note	The user ID is used to copy the SAS Embedded Process configuration files to HDFS.

-epuser ***epuser-name***
specifies the name for the SAS Embedded Process user.

Default	sasep

-epgroup ***epgroup-name***
specifies the name for the SAS Embedded Process group.

Default	sasep

-hostfile ***host-list-filename***
specifies the full path of a file that contains the list of hosts where the SAS Embedded Process is installed, removed, started, stopped, or status is provided.

Default	If you do not specify -hostfile, the sasep-servers.sh script will discover the cluster topology and use the retrieved list of data nodes.
Tip	You can also assign a host list filename to a UNIX variable, **`sas_ephosts_file`**. `export sasep_hosts=/etc/hadoop/conf/slaves`
See	"-hdfsuser *user-id*" on page 63
Example	`-hostfile /etc/hadoop/conf/slaves`

-host <">*host-list*<">
specifies the target host or host list where the SAS Embedded Process is installed, removed, started, stopped, or status is provided.

Default	If you do not specify -host, the sasep-servers.sh script will discover the cluster topology and use the retrieved list of data nodes.
Requirement	If you specify more than one host, the hosts must be enclosed in double quotation marks and separated by spaces.
Tip	You can also assign a list of hosts to a UNIX variable, **`sas_ephosts`**. `export sasep_hosts="server1 server2 server3"`
See	"-hdfsuser *user-id*" on page 63

Example	`-host "server1 server2 server3"` `-host bluesvr`

-epscript *path-to-ep-install-script*
copies and unpacks the SAS Embedded Process install script file to the host.

Restriction	Use this option only with the -add option.
Requirement	You must specify either the full or relative path of the SAS Embedded Process install script, tkindbsrv-9.42-*n*_lax.sh file.
Example	`-epscript /home/hadoop/image/current/tkindbsrv-9.42-1_lax.sh`

-mrscript *path-to-mr-jar-file-script*
copies and unpacks the SAS Hadoop MapReduce JAR files install script on the hosts.

Restriction	Use this option only with the -add option.
Requirement	You must specify either the full or relative path of the SAS Hadoop MapReduce JAR file install script, hadoopmrjars-9.42-*n*_lax.sh file.
Example	`-mrscript /home/hadoop/image/current/hadoopmrjars-9.42-1_lax.sh`

-options "*option-list*"
specifies options that are passed directly to the SAS Embedded Process. The following options can be used.

-trace *trace-level*
specifies what type of trace information is created.

0	no trace log
1	fatal error
2	error with information or data value
3	warning
4	note
5	information as an SQL statement
6	critical and command trace
7	detail trace, lock
8	enter and exit of procedures
9	tedious trace for data types and values
10	trace all information

Default	02
Note	The trace log messages are stored in the MapReduce job log.

-port *port-number*
specifies the TCP port number where the SAS Embedded Process accepts connections.

Default	9261

Requirement	The options in the list must be separated by spaces, and the list must be enclosed in double quotation marks.

-log *filename*
writes the installation output to the specified filename.

-version *apache-version-number*
specifies the Hadoop version of the JAR file that you want to install on the cluster. The *apache-version-number* can be one of the following values.

0.23
installs the SAS Hadoop MapReduce JAR files that are built from Apache Hadoop 0.23 (sas.hadoop.ep.apache023.jar and sas.hadoop.ep.apache023.nls.jar).

1.2
installs the SAS Hadoop MapReduce JAR files that are built from Apache Hadoop 1.2.1 (sas.hadoop.ep.apache121.jar and sas.hadoop.ep.apache121.nls.jar).

2.0
installs the SAS Hadoop MapReduce JAR files that are built from Apache Hadoop 0.2.3 (sas.hadoop.ep.apache023.jar and sas.hadoop.ep.apache023.nls.jar).

2.1
installs the SAS Hadoop MapReduce JAR files that are built from Apache Hadoop 2.0.5 (sas.hadoop.ep.apache205.jar and sas.hadoop.ep.apache205.nls.jar).

Default	If you do not specify the -version option, the sasep.servers.sh script will detect the version of Hadoop that is in use and install the JAR files associated with that version. For more information, see "Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files" on page 55.
Interaction	The -version option overrides the version that is automatically detected by the sasep.servers.sh script.

-getjars
creates a HADOOP_JARS.zip file in the local folder. This ZIP file contains all required client JAR files.

You need to move this ZIP file to your client machine and unpack it. If you want to replace the existing JAR files, move it to the same directory where you previously unpacked the existing JAR files.

See	For more information, see "Copying Hadoop JAR Files to the Client Machine" on page 60.

Starting the SAS Embedded Process

There are three ways to manually start the SAS Embedded Process.

Note: Root authority is required to run the sasep-servers.sh script.

- Run the sasep-servers.sh script with the `-start` option on the master node.

  This starts the SAS Embedded Process on all nodes. For more information about running the sasep-servers.sh script, see "SASEP-SERVERS.SH Syntax" on page 61.

- Run sasep-server-start.sh on a node.

  This starts the SAS Embedded Process on the local node only. The sasep-server-start.sh script is located in the ***SASEPHome*/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/** directory. For more information, see "Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files" on page 55.

- Run the UNIX **service** command on a node.

  This starts the SAS Embedded Process on the local node only. The **service** command calls the init script that is located in the **/etc/init.d** directory. A symbolic link to the init script is created in the **/etc/rc3.d** and **/etc/rc5.d** directories, where **3** and **5** are the run level at which you want the script to be executed.

  Because the SAS Embedded Process init script is registered as a service, the SAS Embedded Process is started automatically when the node is rebooted.

Stopping the SAS Embedded Process

The SAS Embedded Process continues to run until it is manually stopped. The ability to control the SAS Embedded Process on individual nodes could be useful when performing maintenance on an individual node.

There are three ways to stop the SAS Embedded Process.

Note: Root authority is required to run the sasep-servers.sh script.

- Run the sasep-servers.sh script with the **-stop** option from the master node.

  This stops the SAS Embedded Process on all nodes. For more information about running the sasep-servers.sh script, see "SASEP-SERVERS.SH Syntax" on page 61.

- Run sasep-server-stop.sh on a node.

  This stops the SAS Embedded Process on the local node only. The sasep-server-stop.sh script is located in the ***SASEPHome*/SAS/SASTKInDatabaseServerForHadoop/9.42/bin/** directory. For more information, see "Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files" on page 55.

- Run the UNIX **service** command on a node.

  This stops the SAS Embedded Process on the local node only.

Determining the Status of the SAS Embedded Process

You can display the status of the SAS Embedded Process on one node or all nodes. There are three ways to display the status of the SAS Embedded Process.

Note: Root authority is required to run the sasep-servers.sh script.

- Run the sasep-servers.sh script with the **-status** option from the master node.

  This displays the status of the SAS Embedded Process on all nodes. For more information about running the sasep-servers.sh script, see "SASEP-SERVERS.SH Syntax" on page 61.

- Run sasep-server-status.sh from a node.

This displays the status of the SAS Embedded Process on the local node only. The sasep-server-status.sh script is located in the ***SASEPHome*/ SAS/SASTKInDatabaseServerForHadoop/9.42/bin/** directory. For more information, see "Installing the SAS Embedded Process and SAS Hadoop MapReduce JAR Files" on page 55.

- Run the UNIX **`service`** command on a node.

  This displays the status of the SAS Embedded Process on the local node only.

Hadoop Permissions

The person who installs the SAS Embedded Process must have sudo access.

Documentation for Using In-Database Processing in Hadoop

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

- *SAS In-Database Products: User's Guide*
- High-performance procedures in various SAS publications
- *SAS Data Integration Studio: User's Guide*
- SAS/ACCESS Interface to Hadoop and PROC HDMD in *SAS/ACCESS for Relational Databases: Reference*
- *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*
- *SAS Intelligence Platform: Data Administration Guide*
- PROC HADOOP in *Base SAS Procedures Guide*
- FILENAME Statement, Hadoop Access Method in *SAS Statements: Reference*
- *SAS Data Loader for Hadoop: User's Guide*

Chapter 6
Administrator's Guide for Netezza

In-Database Deployment Package for Netezza

Prerequisites

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

The SAS Scoring Accelerator for Netezza and the SAS Embedded Process require a specific version of the Netezza client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

Overview of the In-Database Deployment Package for Netezza

This section describes how to install and configure the in-database deployment package for Netezza (SAS Formats Library for Netezza and SAS Embedded Process).

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 in-database deployment package for Netezza contains the SAS formats library, two pre-complied binaries for utility functions, and the SAS Embedded Process.

The SAS formats library is a run-time library that is installed on your Netezza system. This installation is made 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.

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. These installations are done so that the SAS scoring files created in Netezza can access routines within the SAS Embedded Process run-time libraries.

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 or reinstalling a previous version, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 71.
2. Install the in-database deployment package.

   For more information, see "Installing the SAS Formats Library, Binary Files, and the SAS Embedded Process" on page 73.
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 75.
4. Run the %INDNZ_PUBLISH_COMPILEUDF macro.

   For more information, see"Running the %INDNZ_PUBLISH_COMPILEUDF Macro" on page 77.
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 11, "Configurations for SAS Model Manager," on page 109.

Upgrading from or Reinstalling a Previous Version

Overview of Upgrading from or Reinstalling a Previous Version

You can upgrade from or reinstall a previous version of the SAS Formats Library and binary files, the SAS Embedded Process, or both. See the following topics:

- If you want to upgrade or reinstall a previous version of the SAS Formats Library and binary files, see "Upgrading from or Reinstalling the SAS Formats Library and Binary Files" on page 71.
- If you want to upgrade or reinstall a previous version of the SAS Embedded Process, see "Upgrading from or Reinstalling the SAS Embedded Process" on page 72.

Upgrading from or Reinstalling the SAS Formats Library and Binary Files

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

Note: These steps apply if you want to upgrade from or reinstall only the SAS Formats Library and binary files. If you want to upgrade from or reinstall the SAS Embedded Process, see "Upgrading from or Reinstalling the SAS Embedded Process" on page 72.

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 75.

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 77.

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

   Note: Under the SAS directory, the installer for the SAS Formats Library and binary files and the SAS Embedded Process installer both create a directory under the SAS directory. These directories are named SASFormatsLibraryForNetezza and SASTKInDatabaseServerForNetezza, respectively. If you delete everything under the SAS directory, the SAS Embedded Process, the SAS Formats Library, and the binary files are removed. If you want to remove only one, then you must leave the other directory.

4. If you are also upgrading from or reinstalling the SAS Embedded Process, continue the installation instructions in "Upgrading from or Reinstalling the SAS Embedded Process" on page 72. Otherwise, continue the installation instructions in "Installing the SAS Formats Library, Binary Files, and the SAS Embedded Process" on page 73.

Upgrading from or Reinstalling the SAS Embedded Process

To upgrade from or reinstall a previous version of the SAS Embedded Process, follow these steps.

Note: These steps are for upgrading from or reinstalling only the SAS Embedded Process. If you want to upgrade from or reinstall the SAS Formats Library and binary files, you must follow the steps in "Upgrading from or Reinstalling the SAS Formats Library and Binary Files" on page 71.

1. Check the current installed version of the SAS Embedded Process.

   ```
   nzcm --installed
   ```

2. Enter these commands to unregister and uninstall the SAS Embedded Process.

   ```
   nzcm -u SASTKInDatabaseServerForNetezza
   nzcm -e SASTKInDatabaseServerForNetezza
   ```

3. Navigate to the **/nz/extensions/SASTKInDatabaseServerForNetezza** directory and verify that the directory is empty.

   Note: Under the SAS directory, the installer for the SAS Formats Library and binary files and the SAS Embedded Process installer both create a directory under the SAS directory. These directories are named SASFormatsLibraryForNetezza and SASTKInDatabaseServerForNetezza, respectively. If you delete everything under the SAS directory, the SAS Embedded Process, the SAS Formats Library, and the binary files are removed. If you want to remove only one, then you must leave the other directory.

4. Continue the installation instructions in "Installing the SAS Formats Library, Binary Files, and the SAS Embedded Process" on page 73.

Installing the SAS Formats Library, Binary Files, and the SAS Embedded Process

Moving and 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 ***`SAS-iinstallation-directory`*****`/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 your Netezza system.

   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 reinstall or upgrade, *n* is incremented by 1.

2. After the accelnetzfmt-3.1-*n*_lax.sh file has been transferred to the Netezza machine, log on as the user who owns the Netezza software (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/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.

Moving and Installing the SAS Embedded Process

The SAS Embedded Process is contained in a self-extracting archive file. The self-extracting archive file is located in the ***`SAS-installation-directory`*****`/SASTKInDatabaseServer/9.4/Netezza64bitTwinFin/`** directory.

To move and unpack the self-extracting archive file to create a Netezza cartridge file, follow these steps:

1. Using a method of your choice, transfer the tkindbsrv-9.4_M2-*n*_lax.sh to any directory on the Netezza host machine.

   n is a number that indicates the latest version of the file.

2. After the tkindbsrv-9.4_M2-*n*_lax.sh file has been transferred to the Netezza, log on as the user who owns the Netezza appliance (usually the “nz” ID).

3. If you have a database named SAS_EP, you should rename it.

   When you unpack the self-extracting archive file, a SAS_EP database that contains SAS Embedded Process function is created. The creation of the SAS_EP database overwrites any existing database that is named SAS_EP.

4. Unpack the self-extracting archive file and create a Netezza cartridge file.

   a. Change to the directory where you put the tkindbsrv.sh file.

      ```
      cd path_to_sh_file
      ```

      path_to_sh_file is the location to which you copied the self-extracting archive file in Step 1.

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

      ```
      ./tkindbsrv-9.4_M2-n_lax.sh
      ```

      After the script runs, the tkindbsrv-9.4_M2-*n*_lax.sh file goes away, and the SASTKInDatabaseServerForNetezza-9.4.1.*n*.nzc Netezza cartridge file is created in its place.

5. Use these **nzcm** commands to install and register the sas_ep cartridge.

   ```
   nzcm -i sas_ep
   nzcm -r sas_ep
   ```

   Note: The sas_ep cartridge creates the NZRC database. The NZRC database contains remote controller functions that are required by the SAS Embedded Process. The sas_ep cartridge is available on the Netezza website. For access to the sas_ep cartridge, contact your local Netezza representative.

6. Use these **nzcm** commands to install and register the SAS Embedded Process.

   ```
   nzcm -i SASTKInDatabaseServerForNetezza-9.4.1.n.nzc
   nzcm -r SASTKInDatabaseServerForNetezza
   ```

   Note: The installation of the SAS Embedded Process is dependent on the sas_ep cartridge that is supplied by Netezza.

   For more NZCM commands, see "NZCM Commands for the SAS Embedded Process" on page 74.

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.

Command	Action performed
`nzcm -help`	Displays help for NZCM commands

`nzcm --installed`
`nzcm - i`
	Displays the filename (SASTKInDatabaseServerForNetezza) and the version number that is installed

`nzcm --registered`
`nzcm - r`
	Displays the filename (SASTKInDatabaseServerForNetezza) and the version number that is registered

Command	**Action performed**

`nzcm --unregister SASTKInDatabaseServerForNetezza`
`nzcm -u SASTKInDatabaseServerForNetezza`
	Unregisters the SAS Embedded Process

`nzcm --unregister sas_ep`
`nzcm -u sas_ep`
	Unregisters the sas_ep cartridge
	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 reinstalled. The sas_ep cartridge needs to be unregistered and uninstalled only when Netezza changes the cartridge version.

`nzcm -uninstall SASTKInDatabaseServerForNetezza`
`nzcm -e SASTKInDatabaseServerForNetezza`
	Uninstalls the SAS Embedded Process

`nzcm --uninstall sas_ep`
`nzcm -e sas_ep`
	Uninstalls the sas_ep cartridge
	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 reinstalled. The sas_ep cartridge needs to be unregistered and uninstalled only when Netezza changes the cartridge version.

`nzcm --install SASTKInDatabaseServerForNetezza-9.4.1.`*n*`.nzc`
`nzcm -i SASTKInDatabaseServerForNetezza-9.4.1.`*n*`.nzc`
	Installs the SAS Embedded Process

`nzcm --install sas_ep`
`nzcm -i sas_ep`
	Installs the sas_ep cartridge

`nzcm --register SASTKInDatabaseServerForNetezza`
`nzcm -r SASTKInDatabaseServerForNetezza`
	Registers the SAS Embedded Process

`nzcm --register sas_ep`
`nzcm -register sas_ep`
	Registers the sas_ep cartridge

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 76.

2. Run the %INDNZ_PUBLISH_JAZLIB macro. For more information, see "%INDNZ_PUBLISH_JAZLIB Macro Syntax" on page 77.

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*<'>

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*<'>
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*<'>
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*<'>
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.

Interaction The database that is specified by the %INDNZ_PUBLISH_JAZLIB 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_JAZLIB macro, the default value of SASLIB is used. For more information, see "%INDNZ_PUBLISH_JAZLIB Macro Syntax" on page 77.

Tip The object name for the SAS formats library is **`sas_jazlib`**.

%INDNZ_PUBLISH_JAZLIB Macro Syntax

%INDNZ_PUBLISH_JAZLIB

```
(<DATABASE=database>
  <, 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`**.

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
Tip	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.

Tip	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 70.

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 80.

%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 Run Process" on page 78.

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

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

   For more information, see the "INDCONN Macro Variable" on page 78.

3. Run the %INDNZ_PUBLISH_COMPILEUDF macro. For more information, see "%INDNZ_PUBLISH_COMPILEUDF Macro Syntax" on page 79.

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*<'>

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*<'>
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*<'>
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*<'>
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 79.
	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 82.

%INDNZ_PUBLISH_COMPILEUDF Macro Syntax

%INDNZ_PUBLISH_COMPILEUDF

(<DATABASE=*database-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 78.

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.

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.

Permission Needed	**Authority Required to Grant Permission**	**Examples**
CREATE LIBRARY permission to run the %INDNZ_PUBLISH_JAZLIB macro that publishes the SAS formats library (**`sas_jazlib`** object)	System Administrator or Database Administrator *Note:* If you have SYSADM or DBADM authority, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions.	`GRANT CREATE LIBRARY TO` *`fmtlibpublisherid`*
CREATE FUNCTION permission to run the %INDNZ_PUBLISH_COMPILEUDF macro that publishes the SAS_COMPILEUDF, SAS_DIRECTORYUDF, and the SAS_HEXTOTEXTUDF functions		`GRANT CREATE FUNCTION TO` *`compileudfpublisherid`*

- 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.

Permission Needed	Authority Required to Grant Permission	Examples
EXECUTE permission for the SAS Formats Library	System Administrator or Database Administrator *Note:* If you have SYSADM or DBADM authority, then you have these permissions. Otherwise, contact your database administrator to obtain these permissions.	`GRANT EXECUTE ON SAS_JAZLIB TO` *`scoringorfmtpublisherid`*
EXECUTE permission for the SAS_COMPILEUDF function		`GRANT EXECUTE ON SAS_COMPILEUDF TO` *`scoringorfmtpublisherid`*
EXECUTE permission for the SAS_DIRECTORYUDF function		`GRANT EXECUTE ON SAS_DIRECTORYUDF TO` *`scoringorfmtpublisherid`*
EXECUTE permission for the SAS_HEXTOTEXTUDF function		`GRANT EXECUTE ON SAS_HEXTOTEXTUDF TO` *`scoringorfmtpublisherid`*

CREATE FUNCTION, CREATE TABLE, CREATE TEMP TABLE, and CREATE EXTERNAL TABLE permissions to run the format and scoring publishing macros		`GRANT CREATE FUNCTION TO` *`scoringorfmtpublisherid`*
		`GRANT CREATE TABLE TO` *`scoringorfmtpublisherid`*
		`GRANT CREATE TEMP TABLE TO` *`scoringorfmtpublisherid`*
		`GRANT CREATE EXTERNAL TABLE TO` *`scoringorfmtpublisherid`*
		`GRANT UNFENCED TO` *`scoringorfmtpublisherid`*

- 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 11, "Configurations for SAS Model Manager," on page 109.

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*, located at http://support.sas.com/documentation/onlinedoc/indbtech/index.html.

Chapter 7
Administrator’s Guide for Oracle

In-Database Deployment Package for Oracle

Prerequisites

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

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

Overview of the In-Database Package for Oracle

This section describes how to install and configure the in-database deployment package for Oracle (SAS Embedded Process).

The in-database deployment package 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 the *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.

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

The in-database deployment package for Oracle includes 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 are 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.

Oracle Installation and Configuration

Oracle Installation and Configuration Steps

1. If you are upgrading from or reinstalling a previous release, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 84 before installing the in-database deployment package.
2. Install the in-database deployment package.

   For more information, see "Installing the In-Database Deployment Package for Oracle" on page 84.
3. Create the required users and objects in the Oracle server.

   For more information, see "Creating Users and Objects for the SAS Embedded Process" on page 86.
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 11, "Configurations for SAS Model Manager," on page 109.

Note: If you are installing 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 or Reinstalling a Previous Version

You can upgrade from or reinstall a previous version of the SAS Embedded Process. Before installing the In-Database Deployment Package for Oracle, have the database administrator (DBA) notify the user community that there will be an upgrade of the SAS Embedded Process. The DBA should then alter the availability of the database by restricting access, or by bringing the database down. Then, follow the steps outlined in "Installing the In-Database Deployment Package for Oracle" on page 84.

Installing the In-Database Deployment Package for Oracle

Overview

The in-database deployment package for Oracle is contained in a self-extracting archive file named tkindbsrv-9.4_M2-*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 reinstall or upgrade, *n* is incremented by 1.

The self-extracting archive file is located in the ***`SAS-installation-directory`*`/ SASTKInDatabaseServer/9.4/OracleDatabaseonLinuxx64/`** directory.

Move the SAS Embedded Process Package to the Oracle Server

To move and copy the Oracle in-database deployment package, follow these steps:

1. Using a method of your choice (for example, PSFTP, SFTP, SCP, or FTP), move the tkindbsrv-9.4_M2-*n*_lax.sh file to directory of your choice. It is suggested that you create a SAS directory under your home directory. An example is **`/u01/pochome/SAS`**.

2. Copy the tkindbsrv-9.4_M2-*n*_lax.sh file onto each of the RAC nodes using a method of your choice (for example, DCLI, SFTP, SCP, or FTP).

   Note: This might not be necessary. For RAC environments with a shared Oracle Home, you can also use one of these methods:

   - Copy the extracted directories from a single node.
   - Copy the self-extracting archive file to a directory common to all the nodes.
   - If the file system is not a database file system (DBFS), extract the file in one location for the whole appliance.

Unpack the SAS Embedded Process Files

For each node, log on as the owner user for the Oracle software using a secured shell, such as SSH. Perform the following steps:

1. Change to the directory where the tkindbsrv-9.4_M2-*n*_lax.sh file is located.

2. If necessary, change permissions on the file to enable you to execute the script and write to the directory.

   ```
   chmod +x tkindbsrv-9.4_M2-n_lax.sh
   ```

3. Use this command to unpack the self-extracting archive file.

   ```
   ./tkindbsrv-9.4_M2-n_lax.sh
   ```

   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 the path to your self-extracting archive file. Part of the directory path is shaded to emphasize the different target directories that are used.

   ```
   /path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/bin
   /path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/misc
   /path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/sasexe
   /path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/utilities
   /path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/admin
   /path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/logs
   ```

4. On non-shared Oracle home systems, update the contents of the $ORACLE_HOME/hs/admin/extproc.ora file on each node. On shared Oracle home systems, you can update the file in one location that is accessible by all nodes.

   a. Make a backup of the current extproc.ora file.

   b. Add the following settings to the file making sure to override any previous settings.

      ```
      SET EXTPROC_DLLS=ANY
      ```

```
SET EPPATH=/path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/
SET TKPATH=/path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.4_M2/sasexe
```

Note: Ask your DBA if the ORACLE_HOME environment variable is not set.

5. On non-shared Oracle home systems, update the contents of the $ORACLE_HOME/network/admin/sqlnet.ora file on each node. On shared Oracle home systems, you can update the file in one location that is accessible by all nodes.
   a. Make a backup of the current sqlnet.ora file. If the file does not exist, create one.
   b. Add the following setting to the file.

      ```
      DIAG_ADR_ENABLED=OFF
      ```

Creating Users and Objects for the SAS Embedded Process

After the In-Database Deployment Package 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 ***/path_to_sh_file/*****SAS/SASTKInDatabaseServerForOracle/9.4_M2/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. 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 ***/path_to_sh_file/*****SAS/SASTKInDatabaseServerForOracle/9.4_M2/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 modeltablename TO userid
```

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 11, “Configurations for SAS Model Manager,” on page 109.

Documentation for Using In-Database Processing in Oracle

For information about how to publish SAS scoring models, see the *SAS In-Database Products: User's Guide*, located at http://support.sas.com/documentation/onlinedoc/indbtech/index.html.

Chapter 8

Administrator’s Guide for SAP HANA

In-Database Deployment Package for SAP HANA

Prerequisites

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

The SAS Scoring Accelerator for SAP HANA and the SAS Embedded Process require 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.

Overview of the In-Database Deployment Package for SAP HANA

This section describes how to install and configure the in-database deployment package for SAP HANA (SAS Embedded Process).

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

- Use the %INDHN_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 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*.

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

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.

SAP HANA Installation and Configuration

SAP HANA Installation and Configuration Steps

1. Review the permissions required for installation.

   For more information, see "SAP HANA Permissions" on page 96.
2. Review the number of semaphore arrays configured for the SAP HANA server.

   It is recommended that the SAP HANA server that runs the SAS Embedded Process be configured with a minimum of 1024 to 2048 semaphore arrays. For more information, see "Semaphore Requirements When Using the SAS Embedded Process for SAP HANA" on page 96.
3. Enable the SAP HANA Script Server process as SYSTEM in the SAP HANA Studio.

   The SAP HANA script server process must be enabled to run in the HANA instance. The script server process can be started while the SAP HANA database is already running.

   To start the Script Server, follow these steps:

   a. Open the **Configuration** tab page in the SAP HANA Studio.

   b. Expand the daemon.ini configuration file.

   c. Expand the **scriptserver** section.

   d. Change the **instances** parameter from 0 to 1 at the system level.

      A value of 1 indicates you have enabled the server.

   Note: For more information, see SAP Note 1650957.
4. If you are upgrading from or reinstalling a previous release, follow the instructions in "Upgrading or Reinstalling a Previous Version" on page 91 before installing the in-database deployment package.

5. Install the SAS Embedded Process.

   For more information, see "Installing the In-Database Deployment Package for SAP HANA" on page 91.

6. Install the SASLINK Application Function Library.

   For more information, see "Installing the SASLINK AFL Plugins on the Appliance" on page 93.

7. Import the SAS_EP Stored Procedure.

   For more information, see "Importing the SAS_EP Stored Procedure" on page 94.

8. Verify that the Auxiliary Wrapper Generator and Eraser Procedures are installed in the SAP HANA catalog.

   For more information, see "Auxiliary Wrapper Generator and Eraser Procedures" on page 95.

9. 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 11, "Configurations for SAS Model Manager," on page 109.

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*.

Upgrading or Reinstalling a Previous Version

To upgrade or reinstall a previous version, follow these steps.

1. Log on to the SAP HANA system as root.

   You can use su or sudo to become the root authority.

2. Run the UninstallSASEPFiles.sh file.

   ```
   ./UninstallSASEPFiles.sh
   ```

   The UninstallSASEPFiles.sh file is in the ***/SASEPHome/*** where you copied the tkindbsrv-9.4-*n*_lax.sh self-extracting archive file.

   This script stops the SAS Embedded Process on the server. The script deletes the **/SAS/SASTKInDatabaseServerForSAPHANA** directory and all its contents.

   Note: You can find the location of *SASEPHome* by using the following command:

   ```
   ls -l /usr/local/bin
   ```

3. Reinstall the SAS Embedded Process.

   For more information, see "Installing the In-Database Deployment Package for SAP HANA" on page 91.

Installing the In-Database Deployment Package for SAP HANA

The SAS Embedded Process is contained in a self-extracting archive file. The self-extracting archive file is located in the ***SAS-installation-directory/*** **SASTKInDatabaseServer/9.4/SAPHANAonLinuxx64/** directory.

To install the self-extracting archive file, follow these steps:

1. Using a method of your choice, transfer the tkindbsrv-9.4-*n*_lax.sh file to the target SAS Embedded Process directory on the SAP HANA appliance.

   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 reinstall or upgrade, *n* is incremented by 1.

   This example uses secure copy, and **/SASEPHome/** is the location where you want to install the SAS Embedded Process.

   ```
   scp tkindbsrv-9.4-n_lax.sh username@hana: /SASEPHome/
   ```

   Note: The **SASEPHome** directory requires Read and Execute permissions for the database administrator.

2. After the tkindbsrv-9.4-*n*_lax.sh has been transferred, log on to the SAP HANA server as a superuser.

3. Move to the directory where the self-extracting archive file was downloaded in Step 1.

   ```
   cd /SASEPHome
   ```

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

   ```
   ./tkindbsrv-9.4-n_lax.sh
   ```

   Note: If you receive a "permissions denied" message, check the permissions on the tkindbsrv-9.4-*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. Directories and files of interest are shaded.

   ```
   /SASEPHome/afl_wrapper_eraser.sql
   /SASEPHome/afl_wrapper_generator.sql
   /SASEPHome/InstallSASEPFiles.sh
   /SASEPHome/manifest
   /SASEPHome/mit_unzip.log
   /SASEPHome/saslink.lst
   /SASEPHome/saslink_area.pkg
   /SASEPHome/SAS
   /SASEPHome/SAS_EP_sas.com.tgz
   /SASEPHome/ShowSASEPStatus.sh
   /SASEPHome/ShutdownSASEP.sh
   /SASEPHome/StartupSASEP.sh
   /SASEPHome/tkindbsrv-9.4-n_lax.sh
   /SASEPHome/UninstallSASEPFiles.sh
   ```

   Note that a SAS directory is created where the EP files will be installed. The contents of the **/SASEPHome/SAS/** directories should look similar to these.

   ```
   /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/admin
   /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/bin
   /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/logs
   /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/sasexe
   /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/utilities
   ```

   The InstallSASEPFiles.sh file installs the SAS Embedded Process. The next step explains how to run this file.

   The UninstallSASEPFiles.sh file uninstalls the SAS Embedded Process. The StartupSASEP.sh and ShutdownSASEP.sh files enable you to manually start and stop the SAS Embedded Process. The ShowSASEPStatus.sh file shows the status of the

SAS Embedded Process on each instance. For more information about running these two files, see "Controlling the SAS Embedded Process" on page 95.

5. Use the following command at the UNIX prompt to install the SAS Embedded Process.

   ```
   ./InstallSASEPFiles.sh
   ```

   Note: To execute this script you need root authority. Either use the su command to become the root or use the sudo command to execute this script to install the SAS Embedded Process.

   Note: *-verbose* is on by default and enables you to see all messages generated during the installation process. Specify *-quiet* to suppress messages.

Installing the SASLINK AFL Plugins on the Appliance

The SASLINK Application Function Library (AFL) files are included with the server side components. These files must be copied to the SASLINK AFL plugins directory on the SAP HANA server.

Note: The *SID* referenced in these instructions is the SAP HANA system identifier (for example, **HDB**).

1. If it does not exist, create a plugins directory in the $DIR_SYSEXE directory.

   a. Log on to the SAP HANA server as the root authority.

      You can use one of these commands.

      ```
      su - root
      sudo su -
      ```

   b. Create the plugins directory.

      ```
      mkdir -p /usr/sap/SID/SYS/exe/hdb/plugins
      chown SIDadm:sapsys /usr/sap/SID/SYS/exe/hdb/plugins
      chmod 750 /usr/sap/SID/SYS/exe/hdb/plugins
      exit
      ```

2. Use this command to change the user to the database administrator.

   ```
   su - SIDadm
   ```

3. Stop the SAP HANA database if it is running.

   ```
   HDB stop
   ```

4. If it does not exist, create the SASLINK AFL plugins directory.

   ```
   cdexe
   cd -P ..
   mkdir -p plugins/sas_afl_sdk_saslink_1.00.1.0.0_1
   cdexe
   mkdir -p plugins
   cd -P plugins
   ln -s ../../plugins/sas_afl_sdk_saslink_1.00.1.0.0_1 sas_afl_sdk_saslink
   ```

5. Copy the SASLINK AFL files from the **/*SASEPHome*/ SAS/SASTKInDatabaseServerForSAPHANA/9.4/sasexe** and **/**

***SASEPHome*/SAS/SASTKInDatabaseServerForSAPHANA/9.4/admin**
directories to the SASLINK AFL plugins directory.

```
cdexe
cd plugins/sas_afl_sdk_saslink
cp /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/
   sasexe/libaflsaslink.so .
cp /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/
   admin/saslink.lst .
cp /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/
   admin/saslink_area.pkg .
cp /SASEPHome/SAS/SASTKInDatabaseServerForSAPHANA/9.4/
   admin/manifest .
```

Note: You can find the location of *SASEPHome* by using the following command:

```
ls -l /usr/local/bin
```

6. Restart the SAP HANA database.

```
HDB start
```

Importing the SAS_EP Stored Procedure

The SAS_EP Stored Procedure is used by the %INDHN_RUN_MODEL macro to run the scoring model.

The SAS_EP stored procedure is contained in a delivery unit named SAS_EP_sas.com.tgz. The SAS_EP_sas.com.tgz package was installed in the ***SASEPHome*** directory when the tkindbsrv-9.4-*n*_lax.sh file was unpacked.

Note: You can find the location of *SASEPHome* by using the following command:

```
ls -l /usr/local/bin
```

To import the delivery unit into SAP HANA, follow these steps:

Note: Permissions and roles are required to import the procedure package. For more information, see "SAP HANA Permissions" on page 96.

1. Navigate to the ***SASEPHome*** directory.
2. Copy the SAS_EP_sas.com.tgz package to a client machine on which the SAP HANA Studio client is installed.
3. Import the delivery unit.

   There are several methods of importing the .tgz file. Examples are SAP HANA Studio or the Lifecycle Manager. To import the delivery unit using SAP HANA Studio, follow these steps:

   a. Ensure that you have a connection to the target SAP HANA back end from your local SAP HANA Studio.
   b. Select **File** ⇨ **Import**.
   c. Select **SAP HANA Content** ⇨ **Delivery Unit** and click **Next**.
   d. Select the target system and click **Next**.
   e. In the Import Through Delivery Unit window, select the **Client** check box and select the SAS_EP_sas.com.tgz file.

f. Select the **Overwrite inactive versions** and **Activate object** check boxes.

   The list of objects is displayed under **Object import simulation**.

g. Click **Finish** to import the delivery unit.

Auxiliary Wrapper Generator and Eraser Procedures

Operation of the SASLINK AFL and the SAS Embedded Process requires that the SAP HANA AFL_WRAPPER_GENERATOR and AFL_WRAPPER_ERASER procedures are installed in the SAP HANA catalog. If the procedures are not already installed in the SAP HANA catalogs, then copies of these procedures can be found in the install directory on the server.

```
/SASEPHome/afl_wrapper_generator.sql
/SASEPHome/afl_wrapper_eraser.sql
```

Note: You can find the location of *SASEPHome* by using the following command:

```
ls -l /usr/local/bin
```

To install these procedures, you must execute the SQL scripts, using the SAP HANA Studio, as the SAP HANA user SYSTEM. For more information, see the SAP HANA Predictive Analysis Library (PAL) document.

CAUTION:
If a procedure has already been installed, executing the SQL script causes an error. If you encounter an error, see your SAP HANA database administrator.

Controlling the SAS Embedded Process

The SAS Embedded Process starts when a query is submitted using the SAS_EP procedure. 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 following directory. For more information about these files, see "Installing the In-Database Deployment Package for SAP HANA" on page 91.

```
/SASEPHOME/SAS/SASTKInDatabaseServerForSAPHANA/9.4/
```

Note: You can find the location of *SASEPHome* by using the following command:

```
ls -l /usr/local/bin
```

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.

```
/SASEPHOME/SAS/SASTKInDatabaseServerForSAPHANA/9.4/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.

TIP You do not have to manually restart the SAS Embedded Process. The SAS Embedded Process starts when a query is submitted using the SAS_EP procedure.

StartupSASEP.sh manually starts the SAS Embedded Process. It is designed to be used to manually start the SAS Embedded Process and only after consultation with SAS Technical Support. This script should not be used as part of the normal operation.

Use the following command to manually start the SAS Embedded Process.

```
/SASEPHOME/SAS/SASTKInDatabaseServerForSAPHANA/9.4/StartupSASEP.sh
```

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.

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 "Auxiliary Wrapper Generator and Eraser Procedures" on page 95.

Task	Permission Needed
Unpack the self-extracting archive file	sasowner
Install or uninstall the SAS Embedded Process (run InstallSASEPFiles.sh or UninstallSASEPFiles.sh script)	root authority
Import the SAS_EP procedure package	A user on the SAP HANA server that has at least the CONTENT_ADMIN role or its equivalent
Install AFL plugins (requires starting and stopping the database)	root authority and database administrator
Install an auxiliary procedure generator and eraser	SYSTEM user

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

- EXECUTE ON SYSTEM.afl_wrapper_generator to *userid* | *role*;
- EXECUTE ON SYSTEM.afl_wrapper_eraser to *userid* | *role*;
- AFL__SYS_AFL_SASLINK_AREA_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 11, “Configurations for SAS Model Manager,” on page 109.

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*, located at http://support.sas.com/documentation/onlinedoc/indbtech/index.html.

Chapter 9

Administrator’s Guide for SPD Server

Installation and Configuration Requirements for the SAS Scoring Accelerator for SPD Server

Prerequisites

The SAS Scoring Accelerator for SPD Server requires SAS Scalable Performance Data Server 5.1 and SAS 9.4.

If you have a model that was produced by SAS Enterprise Miner, an active SPD Server, and a license for the SAS Scoring Accelerator for SPD Server, you have everything that you need to run scoring models in the SPD Server. Installation of an in-database deployment package is not required.

SPD Server Permissions

You must have permissions for the domains that you specify in the INDCONN and INDDATA macro variables when you execute the publish and run macros.

You also need regular Read, Write, and Alter permissions when writing files to the OUTDIR directory in the %INDSP_RUN_MODEL macro.

Where to Go from Here

For more information about using the SAS Scoring Accelerator for SPD Server, see the *SAS In-Database Products: User's Guide*.

Chapter 10

Administrator's Guide for Teradata

In-Database Deployment Package for Teradata

Prerequisites

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

The SAS Scoring Accelerator for Teradata requires a specific version of the Teradata client and server environment. For more information, see the SAS Foundation system requirements documentation for your operating environment.

If you are using Teradata 13.10, 14.00, or 14.10, you must run DIPGLOP from the Teradata DIP utility before you install the SAS Embedded Process. DIPGLOP installs the DBCEXTENSION.ServerControl procedure. This procedure is used to stop and shut down the SAS Embedded Process. DIPGLOP is not required for Teradata 15.00 or later.

The SAS Embedded Process installation requires approximately 200MB of disk space in the /opt file system on each Teradata TPA node.

Overview of the In-Database Deployment Package for Teradata

This section describes how to install and configure the in-database deployment package for Teradata (SAS Formats Library for Teradata and SAS Embedded Process). The in-database deployment packages 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.
- Use the %INDTD_PUBLISH_MODEL scoring publishing macro to publish scoring model files or functions inside the database.
- 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 DS2 Language Reference*.
- 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*.

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

The in-database deployment package for Teradata includes the SAS formats library and the SAS Embedded Process.

The SAS formats library is a run-time library that is installed 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.

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. These installations are done so that the SAS scoring files created in Teradata can access to routines within its run-time libraries.

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

Note: In addition to the in-database deployment package for Teradata, a set of SAS Embedded Process functions must be installed in the Teradata database. The SAS Embedded Process functions package is downloadable from Teradata.For more information, see "Installing the SAS Embedded Process Support Functions" on page 106.

Teradata Installation and Configuration

Teradata Installation and Configuration Steps

1. If you are upgrading from or reinstalling a previous version, follow the instructions in "Upgrading from or Reinstalling a Previous Version" on page 103.
2. Install the in-database deployment package.

For more information, see "Installing the SAS Formats Library and the SAS Embedded Process" on page 104.

3. Install the SAS Embedded Process support functions.

   For more information, see "Installing the SAS Embedded Process Support Functions" on page 106.

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 11, "Configurations for SAS Model Manager," on page 109.

Note: If you are installing 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 or Reinstalling a Previous Version

To upgrade from or reinstall a previous version of the SAS Formats Library, the SAS Embedded Process, or both, 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 -q -a" | 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 version of the formats library is currently installed, run this command:

     ```
     psh "rpm -q -a" | grep acc
     ```

     If a previous version is installed, a result similar to this is displayed. The version number might be different.

     ```
     accelterafmt-2.1-1
     ```

   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 the SAS Embedded Process" on page 104.

2. Run this command to check the current installed version of the SAS Embedded Process.

   ```
   psh "rpm -qa | grep tkindbsrv"
   ```

   If a previous version is installed, a result similar to this is displayed. The version number might be different.

   ```
   tkindbsrv-9.42_M1-2
   ```

   If the SAS Embedded Process 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 the SAS Embedded Process" on page 104.

3. If a version of the SAS formats library, the SAS Embedded Process, or both is being installed that has a name that is different from the library that was previously installed, then follow these steps. An example would be accelterafmt-3.1-1 replacing jazxfbrs-9.2-1.6 or tkindbsrv-9.42_M1-2 replacing tkindbsrv-9.35-1.

   a. If you are upgrading from or reinstalling the SAS Formats Library, 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.

      Note: If you are upgrading or reinstalling only the SAS Embedded Process (tkindbsrv.rpm file), you do not need to shut down the database. You do need to shut down the SAS Embedded Process. For more information, see "Controlling the SAS Embedded Process" on page 106.

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

      - To remove the package from all nodes concurrently, run this command:

        ```
        psh "rpm -e package-name"
        ```

        package-name is either jazxfbrs.9.*version*, accelterafmt-2.*version*, or tkindbsrv-9.35-*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*, accelterafmt-2.*version*, or tkindbsrv-9.35-*version*.

4. (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*, accelterafmt-2.*version*, or tkindbsrv-9.35-*version*.

   The SAS Formats Library or the SAS Embedded Process should not appear on any node.

Installing the SAS Formats Library and the SAS Embedded Process

Moving the SAS Formats Library and the SAS Embedded Process Packages to the Server Machine

1. Locate the in-database deployment package files, accelterafmt-3.1-*n*.x86_64.rpm and tkindbsrv-9.42_M1-*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. Each time you reinstall or upgrade, *n* is incremented by 1.

The accelterafmt-3.1-*n*.x86_64.rpm file is located in the ***SAS-installation-directory*/SASFormatsLibraryforTeradata/3.1/TeradataonLinux/** directory. The tkindbsrv-9.42_M1-*n*.x86_64.rpm file is located in the ***SAS-installation-directory*/SASTKInDatabaseServer/9.4/TeradataonLinux/** directory.

2. Put the package files on your Teradata database server in a location where it is both read and write accessible.

   The package files 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.

Note: The SAS Embedded Process might require a later release of Teradata than function-based scoring. Please refer to the system requirements documentation.

Installing the SAS Formats Library and the SAS Embedded Process 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. 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. Move the SAS Formats Library and the SAS Embedded Process packages to your server machine where they can be accessed from at least one of the Teradata nodes. For more information, see “Moving the SAS Formats Library and the SAS Embedded Process Packages to the Server Machine” on page 104.

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* or tkindbsrv-9.42_M1-*n* is displayed. *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 reinstall or upgrade, *n* is incremented by 1.

   Alternatively, you can manually verify that the installation is successful by running these commands from the shell prompt.

   ```
   psh "rpm -q -a" | grep accelterafmt
   psh "rpm -q -a" | grep tkindbsrv
   ```

6. (Optional) Start the server so that all database processes can load the new version of the library and the SAS Embedded Process.

   ```
   /etc/init.d/tpa start
   ```

   Note: If you are upgrading from or reinstalling a previous version of the SAS Formats Library for Teradata, the database was shut down in preparation for this procedure, and no other database maintenance needs to be performed at this time, you should start the database.

Installing the SAS Embedded Process Support Functions

The SAS Embedded Process support function package (sasepfunc) includes stored procedures that generate SQL to interface with the SAS Embedded Process and functions that load the SAS program and other run-time control information into shared memory. The SAS Embedded Process support functions setup script creates the SAS_SYSFNLIB database and the SAS Embedded Process interface fast path functions in TD_SYSFNLIB.

The SAS Embedded Process support function package is available from the Teradata Software Server. For access to the package, contact your local Teradata account representative or the Teradata consultant supporting your SAS and Teradata integration activities.

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
Provides the status of the SAS Embedded Process.	CALL DBCEXTENSION.SERVERCONTROL ('status', :A); * CALL DBCEXTENSION.SERVERCONTROL ('SAS', 'status', :A); ** CALL SYSLIB.SERVERCONTROL ('SAS', 'status', :A); ***
Shuts down the SAS Embedded Process. *Note:* You cannot shut down until all queries are complete.	CALL DBCEXTENSION.SERVERCONTROL ('shutdown', :A); * CALL DBCEXTENSION.SERVERCONTROL ('SAS', 'shutdown', :A); ** CALL SYSLIB.SERVERCONTROL ('SAS', 'shutdown', :A); ***
Stops new queries from being started. Queries that are currently running continue to run until they are complete.	CALL DBCEXTENSION.SERVERCONTROL ('disable', :A); * CALL DBCEXTENSION.SERVERCONTROL ('SAS', 'disable', :A); ** CALL SYSLIB.SERVERCONTROL ('SAS', 'disable', :A); ***

Action performed	Command

Enables new queries to start running.	CALL DBCEXTENSION.SERVERCONTROL ('enable', :A);*
	 CALL DBCEXTENSION.SERVERCONTROL ('SAS', 'enable', :A);**
	 CALL SYSLIB.SERVERCONTROL ('SAS', 'enable', :A); ***

* For Teradata 13.10 and 14.00 only. Note that the Cmd parameter (for example, 'status') must be lowercase.

** For Teradata 14.10 only. Note that the Languagename parameter, 'SAS', is required and must be uppercase. The Cmd parameter (for example, 'status'), must be lowercase.

*** For Teradata 15 only. Note that the Languagename parameter, 'SAS', is required and must be uppercase. The Cmd parameter (for example, 'status'), must be lowercase.

Teradata Permissions

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 11, “Configurations for SAS Model Manager,” on page 109.

Documentation for Using In-Database Processing in Teradata

- *SAS In-Database Products: User's Guide*
- *SAS DS2 Language Reference*
- *SAS Data Quality Accelerator for Teradata: User's Guide*

Chapter 11

Configurations for SAS Model Manager

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 13.1). 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.

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:

1. Create a separate database where the tables can be stored.
2. Set the user access permissions for the database.
   a. GRANT CREATE, DROP, EXECUTE, and ALTER permissions for functions and procedures.

      For more information about permissions for the specific databases, see the following topics:

      - "DB2 Permissions" on page 28
      - "Greenplum Permissions" on page 48
      - "Netezza Permissions" on page 80
      - "Oracle Permissions" on page 87
      - "SAP HANA Permissions" on page 96
      - "Teradata Permissions" on page 107

b. GRANT CREATE and DROP permissions for tables. With these permissions, users can validate the scoring results when publishing a 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 he 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:

1. Create a separate database where the tables can be stored.
2. Set the user access permissions for the database.
   a. GRANT CREATE, DROP, EXECUTE, and ALTER permissions for functions and procedures.

      For more information about permissions for the specific databases, see the following topics:

      - "DB2 Permissions" on page 28
      - "Greenplum Permissions" on page 48
      - "Netezza Permissions" on page 80
      - "Teradata Permissions" on page 107

   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.

3. Navigate to the **\*sasinstalldir*\SASModelManagerInDatabaseScoringScripts\13.1\Utilities** directory to find the Create Tables and Drop Tables scripts for your database. Then, perform the following 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.

4. 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*\Lev#\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*\Lev#\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 112.

5. 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** ⇨ **Downloads** ⇨ **Teradta JDBC Driver**.

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:

1. Copy the Hadoop core and common Hadoop JAR files to the client machine. For more information, see "Copying Hadoop JAR Files to the Client Machine" on page 60.

2. 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.

3. Grant users Write access permission to the HDFS directory.

4. Add these lines of code to the autoexec_usermods.sas file that is located in the Windows directory**\*SAS-configuration-directory*\Lev#\SASApp \WorkspaceServer\**:

   ```
   %let HADOOP_Config = %str(merged-configuration-filepath);
   %let HADOOP_Auth = Kerberos or blank;
   ```

   UNIX Specifics
   The location of the autoexec_usermods.sas file for UNIX is **/*SAS-confirguration-directory*/Lev#/SASApp/WorkspaceServer/**.

   HADOOP_Config is used to replace the use of a merged configuration file. The configuration files are copied to a location on the client machine and the HADOOP_Config variable is set to that location. If your Hadoop server is configured with Kerberos, set the HADOOP_Auth variable to Kerberos. Otherwise, leave it blank. For more information about the merged configuration file, see "How to Merge Configuration File Properties" on page 58.

5. (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\**:

   ```
   -AUTOEXEC '\SAS-confirguration-directory\Lev#\SASApp\WorkspaceServer\
   autoexec_usermods.sas'
   ```

   UNIX Specifics
   The location of the sasv9.cfg file for UNIX is **/SASHome/SASFoundation/ 9.4/**.

Recommended Reading

Here is the recommended reading list for this title:

- *SAS/ACCESS for Relational Databases: Reference*
- *SAS Data Loader for Hadoop: User's Guide*
- *SAS Data Quality Accelerator for Teradata: User's Guide*
- *SAS DS2 Language Reference*
- *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: User's Guide*

Index