THE POWER TO KNOW.

SAS® 9.3 In-Database Products

Administrator's Guide

Second Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 In-Database Products: Administrator's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.3 In-Database Products: Administrator's Guide, Second Edition

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

Electronic book 4, June 2012

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

Contents

Recommended Reading . . . . . *v*

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

Chapter 2 • Administrator's Guide for Aster nCluster **3**
In-Database Deployment Package for Aster nCluster 3

Chapter 3 • Administrator's Guide for DB2 . . . . . **9**
In-Database Deployment Package for DB2 9

Chapter 4 • Administrator's Guide to Greenplum **31**
In-Database Deployment Package for Greenplum 31

Chapter 5 • Administrator's Guide for Netezza **41**
In-Database Deployment Package for Netezza 41

Chapter 6 • Administrator's Guide for Oracle **51**
In-Database Deployment Package for Oracle 51

Chapter 7 • Administrator's Guide for Teradata **57**
In-Database Deployment Package for Teradata 57

Chapter 8 • Configurations for SAS Model Manager **63**
Preparing a Database for Use with SAS Model Manager 63

Index **67**

Recommended Reading

Here is the recommended reading list for this title:

- *SAS/ACCESS for Relational Databases: Reference*
- *SAS In-Database Products: User's Guide*
- *SAS Model Manager: User's Guide*

For a complete list of SAS publications, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

Chapter 1

Introduction to the Administrator's Guide

SAS In-Database Products 1

What Is Covered in This Document? 1

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, and formatted SQL queries inside the database. When using conventional processing, all rows of data are returned from the database to SAS.

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

- SAS/ACCESS Interface to Aster *n*Cluster, SAS/ACCESS Interface to DB2, SAS/ACCESS Interface to Greenplum, SAS/ACCESS Interface to Oracle, SAS/ACCESS Interface to Netezza, 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 the SAS Scoring Accelerator.

- SAS Scoring Accelerator for Aster *n*Cluster, SAS Scoring Accelerator for DB2, SAS Scoring Accelerator for Greenplum, SAS Scoring Accelerator for Netezza, SAS Scoring Accelerator for Oracle, and SAS Scoring Accelerator for Teradata
- SAS Analytics Accelerator for Teradata
- SAS Model Manager In-Database Scoring Scripts

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 and SAS Scoring Accelerator for your database. 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 *n*Cluster 9.31
- SAS Formats Library for DB2 2.1
- SAS Embedded Process for DB2 9.31
- SAS Formats Library for Greenplum 2.2
- SAS Formats Library for Netezza 2.1
- SAS Embedded Process for Oracle 9.31
- SAS Formats Library for Teradata 2.2
- SAS Embedded Process for Teradata 9.31

Additional configuration tasks are needed if you want to use SAS Model Manager for in-database scoring with DB2, Greenplum, Netezza, or Teradata. This document provides detailed instructions for configuring a database for use with SAS Model Manager.

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

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

This document is divided by database management systems.

Chapter 2

Administrator's Guide for Aster *n*Cluster

In-Database Deployment Package for Aster nCluster **3**
Prerequisites 3
Overview of the In-Database Deployment Package for Aster nCluster 3
Aster nCluster Installation and Configuration Steps 4
Upgrading from or Reinstalling a Previous Version 4
Installing the In-Database Deployment Package Binary Files for Aster nCluster . . . 4
Validating the Publishing of the SAS_SCORE( ) and the SAS_PUT( ) Functions 6
Aster nCluster Permissions 6
Documentation for Publishing SAS Formats and Scoring Models in Aster nCluster 7

In-Database Deployment Package for Aster *n*Cluster

Prerequisites

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

Overview of the In-Database Deployment Package for Aster nCluster

This section describes how to install and configure the in-database deployment package for Aster *n*Cluster (SAS Embedded Process 9.31).

The in-database deployment package for Aster *n*Cluster 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.

The scoring and format publishing macros are included in the SAS/ACCESS Interface to Aster *n*Cluster. 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 *n*Cluster includes the SAS Embedded Process. The SAS Embedded Process is a SAS server process that runs within Aster *n*Cluster to read and write data. The SAS Embedded Process contains macros, run-time

libraries, and other software that is installed on your Aster *n*Cluster system so that the SAS_SCORE( ) and the SAS_PUT( ) functions can access the routines within its run-time libraries.

Aster nCluster 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 nCluster" on page 4.

Upgrading from or Reinstalling a Previous Version

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

1. Log in 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.

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

Installing the In-Database Deployment Package Binary Files for Aster nCluster

The in-database deployment package binary files for Aster *n*Cluster are contained in a self-extracting archive file named tkindbsrv-9.31-*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-install-directory*/SASTKInDatabaseServer/9.31/AsternClusteronLinuxx64/** directory.

To install the in-database deployment package binary files for Aster *n*Cluster, 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-9.31-*n*_lax.sh, execute tkindbsrv-9.31-*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-install-directory/SASTKInDatabaseServer/9.31/AsternClusteronLinuxx64/
   ```

2. Log in 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-9.31-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-9.31-n_lax.sh
   ```

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

   ```
   ./tkindbsrv-9.31-n_lax.sh
   ```

9. Change to the directory where SAS is installed.

   ```
   cd /home/beehive/partner/SAS/SASTKInDatabaseServerForAster/9.31-n/sasexe
   ```

10. 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 *n*Cluster, it is recommended that you remove the existing SQL/MR functions before installing the new ones by entering 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
```

11. Exit the ACT tool.

```
\q
```

12. 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/
      9.31-n/sasexe/tkeastmr.so"; \
   ssh $ip "ls -al /home/beehive/partner/SAS/SASTKInDatabaseServerForAster/
      9.31-n/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 *n*Cluster 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 nCluster Permissions

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

For Aster *n*Cluster 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 *n*Cluster 4.6, 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 Publishing SAS Formats and Scoring Models in Aster nCluster

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 . . . . . **9**
Prerequisites 9
Overview of the In-Database Deployment Package for DB2 9
Publishing Process in DB2 10
DB2 Installation and Configuration Steps 10
Upgrading from or Reinstalling a Previous Version 11
Installing the SAS Formats Library, Binary Files, and SAS Embedded Process . . . 14
Running the %INDB2_PUBLISH_COMPILEUDF Macro 20
Running the %INDB2_PUBLISH_DELETEUDF Macro 24
Validating the Publishing of SAS_COMPILEUDF and SAS_DELETEUDF Functions and Global Variables 26
DB2 Permissions 27
Documentation for Publishing SAS Formats or Scoring Models in DB2 29

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.

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 2.1 and SAS Embedded Process 9.31).

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.

The format and scoring publishing macros are included in SAS/ACCESS Interface to DB2. 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 publishing macros. Starting in December 2011, 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.

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 and SAS 9.3*, located at **`http://support.sas.com/techsup/technote/ts800.pdf`**.

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 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 8, "Configurations for SAS Model Manager," on page 63.

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 11.
- 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;
%indb2pc;
%indb2_publish_compileudf(action=drop, db2path=/db2/9.3/sqllib,
   compiler_path=/usr/vac/bin);
%indb2pd;
%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 in 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
$db 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 11.

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-install-directory*/SASFormatsLibraryForDB2/2.1/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-2.1-*n*_r64.sh
  - Linux(x86_64): acceldb2fmt-2.1-*n*_lax.sh

  The file does not have to be downloaded to a specific location, but 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-install-directory*/SASTKInDatabaseServer/9.31/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-9.31-*n*_r64.sh
  - Linux(x86_64): tkindbsrv-9.31-*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-2.1-*n*_lax.sh or acceldb2fmt-2.1-*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 in 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-2.1-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-2.1-*n*_lax.sh or acceldb2fmt-2.1-*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/SASFormatsLibraryForDB2/2.1-n/bin/
      InstallAccelDB2Fmt.sh

   /path_to_sh_file/SAS/SASFormatsLibraryForDB2/2.1-n/bin/CopySASFiles.sh

   /path_to_sh_file/SAS/SASFormatsLibraryForDB2/2.1-n/lib/SAS_CompileUDF

   /path_to_sh_file/SAS/SASFormatsLibraryForDB2/2.1-n/lib/SAS_DeleteUDF

   /path_to_sh_file/SAS/SASFormatsLibraryForDB2/2.1-n/lib/libjazxfbrs.so

   /path_to_sh_file/SAS/SASFormatsLibraryForDB2/2.1 ->2.1-n
   ```
6. Use the following command to place the files in the DB2 instance:

   ```
   $ path_to_sh_file/SAS/SASFormatsLibraryForDB2/2.1-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 command. Otherwise, the CopySASFiles.sh command 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.9.31-*n*_lax.sh or tkindbsrv9.31-*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 in 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-9.31-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-9.31-*n*_lax.sh or tkindbsrv-9.31-*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/SASTKInDatabaseServerForDB2/9.31-n/bin

/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.31-n/misc

/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.31-n/sasexe

/db2instancepath/SAS/SASTKInDatabaseServerForDB2/9.31-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/SASTKInDatabaseServerForDB2/9.31-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/SASTKInDatabaseServerForDB2/9.31-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.

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 will 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 27.

%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 22.

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

2. Start SAS 9.3 and submit the following commands in the Enhanced Editor or Program Editor:

   ```
   %indb2pc;
   %let indconn = server=yourserver user=youruserid password=yourpwd
      database=yourdb schema=saslib;
   ```

   For more information, see "%INDB2PC Macro" on page 21 and "INDCONN Macro Variable" on page 21.

3. Run the %INDB2_PUBLISH_COMPILEUDF macro. For more information, see "%INDB2_PUBLISH_COMPILEUDF Macro Syntax" on page 22.

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 26.

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.

%INDB2PC Macro

The %INDB2PC macro is an autocall library that initializes the %INDB2_PUBLISH_COMPILEUDF macro.

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 10.

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: You can 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 gcc 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 27.

%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 9.3 and submit the following commands in the Enhanced Editor or Program Editor.

   ```
   %indb2pd;
   %let indconn = server=yourserver user=youruserid password=yourpwd
      database=yourdb schema=saslib;
   ```

   For more information, see "%INDB2PD Macro" on page 25 and "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 26.

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.

%INDB2PD Macro

The %INDB2PD macro is an autocall library that initializes the %INDB2_PUBLISH_DELETEUDF macro.

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 10.

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: You can 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 CREATE TABLE 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 8, "Configurations for SAS Model Manager," on page 63.

Documentation for Publishing SAS Formats or Scoring Models 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 **31**
Prerequisites 31
Overview of the In-Database Deployment Package for Greenplum 31
Function Publishing Process in Greenplum 32
Greenplum Installation and Configuration Steps 32
Upgrading from or Reinstalling a Previous Version 33
Moving and Unpacking the SAS Formats Library and Binary Files 33
Running the %INDGP_PUBLISH_COMPILEUDF Macro 35
Validating the Publishing of the SAS_COMPILEUDF, SAS_COPYUDF, SAS_DIRECTORYUDF, and SAS_DEHEXUDF Functions 38
Greenplum Permissions 39
Documentation for Publishing SAS Formats and Scoring Models in Greenplum . . 39

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.

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 2.2).

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 model functions inside the database.

The format and scoring publishing macros are included in the SAS/ACCESS Interface to Greenplum. 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 publishing macros.

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

Function Publishing Process in Greenplum

To publish the scoring model functions and the SAS_PUT( ) function to a Greenplum database, the publishing macros perform the following tasks:

- Create and transfer the source files to the Greenplum server.

  The files are transferred through database tables. 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 a temporary directory on the database server, the source files are converted back to text.

- Compile those source files into object files using the appropriate compiler for the Greenplum system.
- Link with the SAS formats library.
- Copy the shared object files to ***`full-path-to-pkglibdir`*`/SAS`**. The object files are loaded when the scoring model functions are called.
- Register the format and scoring model functions in Greenplum with those object files. If an existing format or scoring model function is replaced, the publishing macros replace the obsolete object file upon successful compilation and publication of the new format or scoring model function.

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. Move and unpack the SAS formats library and binary files for the publishing macro.

   For more information, see "Moving and Unpacking the SAS Formats Library and Binary Files" on page 33.

3. Run the %INDGP_PUBLISH_COMPILEUDF macro.

   For more information, see "Running the %INDGP_PUBLISH_COMPILEUDF Macro" on page 35.

Upgrading from or Reinstalling a Previous Version

If you are upgrading from or reinstalling a previous version, follow the instructions in "Greenplum Installation and Configuration Steps." However, you need to run the %INDGP_PUBLISH_COMPILEUDF macro twice.

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. To do this, run the %INDGP_PUBLISH_COMPILEUDF macro with ACTION=DROP. Then rerun the %INDGP_PUBLISH_COMPILEUDF macro with ACTION=CREATE. For more information, see "Running the %INDGP_PUBLISH_COMPILEUDF Macro" on page 35.

Moving and Unpacking 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-install-directory*/SASFormatsLibraryforGreenplum/2.2/GreenplumonLinux64/** directory.

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

1. Using a method of your choice, transfer the accelgplmfmt-2.2-*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 accelgplmfmt-2.2-*n*_lax.sh has been transferred, log in to the Greenplum master node.

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-2.2-1_lax.sh
   ```

   Note: If you receive a "permissions denied" message, check the permissions on the accelgplmfmt-2.2-*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_tar_file* is the location to which you copied the self-extracting archive file.

   ```
   /path_to_tar_file/SAS/SASFormatsLibraryForGreenplum/2.2-1/bin/
       InstallAccelGplmFmt.sh
   /path_to_tar_file/SAS/SASFormatsLibraryForGreenplum/2.2-1/bin/
       CopySASFiles.sh
   /path_to_tar_file/SAS/SASFormatsLibraryForGreenplum/2.2-1/lib/
       SAS_CompileUDF.so
   /path_to_tar_file/SAS/SASFormatsLibraryForGreenplum/2.2-1/lib/
       libjazxfbrs.so
   ```

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

```
./path_to_tar_file/SAS/SASFormatsLibraryForGreenplum/2.2-1/bin/
    CopySASFiles.sh
```

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

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

```
$ pg_config --pkglibdir
```

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, including SAS_CompileUDF.so, libjazxfbrs.so, and the binary files for the already published functions, to the new nodes using a method of your choice.

   In addition, you must follow these steps from the master node to create the symbolic links to the SAS formats library for Greenplum (libjazfbrs.so).

The symbolic links are created where the library was loaded on each node in the database array including the master and all segments.

a. Use the following command to determine the full path to where the library was loaded.

```
$ pg_config --libdir
```

This is the path where the symbolic link is created.

b. Use the following command to determine the SAS In-Database shared library deployment path.

```
$ pg_config --pkglibdir
```

This is the path that is linked to and where the SAS formats library is deployed.

c. Use the following command to create the symbolic link on the master node.

```
$ ln -s path-from-pg_config --pkglibdir/SAS/libjazxfbrs.so
   path-from-pg_config --libdir/libjazxfbrs.so
```

Use the value from Step 6b for *path-from-pg_config --pkglibdir*. Use the value from Step 6a for *path-from-pg_config --libdir*.

d. Use the following commands to connect to each of the segment nodes and create the symbolic links on each of the nodes.

```
/* Use this command from the master node to connect to each segment node */
$ ssh <segment nodename>
/* Use this commmand on each segment node to create the link */
$ ln -s path-from-pg_config --pkglibdir/SAS/libjazxfbrs.so
    path-from-pg_config --libdir/libjazxfbrs.so
```

Use the value from Step 6b for *path-from-pg_config --pkglibdir*. Use the value from Step 6a for *path-from-pg_config --libdir*.

To verify that the link is created correctly, go to the directory that results from running the **pg_config --libdir** command and list libjazxfbrs.so.

Running the %INDGP_PUBLISH_COMPILEUDF Macro

Overview of the %INDGP_PUBLISH_COMPILEUDF Macro

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. You can use the **`pg_config --pkglibdir`** command to determine the *full-path-to-pkglibdir* directory.

- 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.

  For more information about the file transfer process, see "Function Publishing Process in Greenplum" on page 32.

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

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 37.

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

2. Start SAS 9.3 and submit the following commands in the Enhanced Editor or Program Editor:

   ```
   %indgppc;
   %let indconn = user=youruserid password=yourpwd dsn=yourdsn;
   /* You can use server=yourserver database=yourdb instead of dsn=yourdsn */
   ```

   For more information, see "%INDGPPC Macro" on page 36 and "INDCONN Macro Variable" on page 36.

3. Run the %INDGP_PUBLISH_COMPILEUDF macro. For more information, see "%INDGP_PUBLISH_COMPILEUDF Macro Syntax" on page 37.

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 38.

%INDGPPC Macro

The %INDGPPC macro is an autocall library that initializes the %INDGP_PUBLISH_COMPILEUDF macro.

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*

USER=<'>*userid*<'> PASSWORD=<'>*password*<'> SERVER=<'>*server*<'> DATABASE=<'>*database*<'>

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: You can 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

Note: The default port that is specified by Greenplum is 5432.

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 **`pg_config --pkglibdir`** command to determine the name of the *full-path-to-pkglibdir* directory.

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.

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.

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
```

Greenplum Permissions

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

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 8, “Configurations for SAS Model Manager,” on page 63.

Documentation for Publishing SAS Formats and Scoring Models 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`**.

Chapter 5

Administrator's Guide for Netezza

In-Database Deployment Package for Netezza **41**
Prerequisites 41
Overview of the In-Database Deployment Package for Netezza 41
Function Publishing Process in Netezza 42
Netezza Installation and Configuration Steps 42
Upgrading from or Reinstalling a Previous Version 43
Installing the SAS Formats Library and Binary Files 43
Running the %INDNZ_PUBLISH_JAZLIB Macro 44
Running the %INDNZ_PUBLISH_COMPILEUDF Macro 46
Netezza Permissions 48
Documentation for Publishing SAS Formats and Scoring Models in Netezza 50

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.

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 2.1).

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.

The format and scoring publishing macros are included in SAS/ACCESS Interface to Netezza. 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 and two additional publishing macros.

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

Function Publishing Process in Netezza

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

- Create and transfer the files, using the Netezza External Table interface, to the Netezza server.

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

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 43.
2. Move and unpack the SAS formats library and binary files for the SAS_COMPILEUDF function.

   For more information, see "Installing the SAS Formats Library and Binary Files" on page 43.
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 44.
4. Run the %INDNZ_PUBLISH_COMPILEUDF macro.

   For more information, see"Running the %INDNZ_PUBLISH_COMPILEUDF Macro" on page 46.
5. 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 8, "Configurations for SAS Model Manager," on page 63.

Upgrading from or Reinstalling a Previous Version

If you are upgrading from or reinstalling a previous version of the SAS formats library and binary files, follow these steps.

1. Run the %INDNZ_PUBLISH_JAZLIB macro with ACTION=DROP to remove the SAS formats library as an object.

   For more information, see "Running the %INDNZ_PUBLISH_JAZLIB Macro" on page 44.

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 46.

3. Navigate to the **`/nz/extensions`** directory and delete the **`SAS`** directory under **`/nz/exensions`**.

   Note: Under the SAS directory, the installer for the SAS Formats Library and binary files creates a directory under the SAS directory. This directory is named SASFormatsLibraryForNetezza. If you delete everything under the SAS directory, the SAS Formats Library and the binary files are removed.

4. Continue the installation instructions in "Installing the SAS Formats Library and Binary Files" on page 43.

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-install-directory`*`/SASFormatsLibraryforNetezza/2.1/Netezza32bitTwinFin/`** directory.

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

1. Using a method of your choice, transfer the accelnetzfmt-2.1-*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-2.1-*n*_lax.sh file has been transferred to the Netezza machine, log in as the user who owns the Netezza software (usually the "nz" ID).

3. Use the following commands at the UNIX prompt to unpack the TAR file.

   ```
   mkdir -p /nz/extensions
   chmod 755 /nz/extensions
   cd /nz/extensions
   chmod 755 accelnetzfmt-2.1-n_lax.sh
   path_to_self-extracting_tar_file/accelnetzfmt-2.1-n_lax.sh
   ```

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

```
/nz/extensions/SAS/SASFormatsLibraryForNetezza/2.1-n/bin/InstallAccelNetzFmt.sh
/nz/extensions/SAS/SASFormatsLibraryForNetezza/2.1-n/lib/SAS_CompileUDF.o_diab_ppc
/nz/extensions/SAS/SASFormatsLibraryForNetezza/2.1-n/lib/SAS_CompileUDF.o_x86
/nz/extensions/SAS/SASFormatsLibraryForNetezza/2.1-n/lib/libjazxfbrs_diab_ppc.a
/nz/extensions/SAS/SASFormatsLibraryForNetezza/2.1-n/lib/libjazxfbrs_x86.a
```

There also is a symbolic link such that **`/nz/extensions/SAS/SASFormatsLibraryForNetezza/2.1`** points to the latest version.

Running the %INDNZ_PUBLISH_JAZLIB Macro

Overview of Publishing the SAS Formats Library

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

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

%INDNZ_PUBLISH_JAZLIB Macro Run Process

To run the %INDNZ_PUBLISH_JAZLIB macro follow these steps:

1. Start SAS 9.3 and submit the following commands in the Enhanced Editor or Program Editor:

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

   For more information, see "%INDNZPJ Macro" on page 44 and "INDCONN Macro Variable" on page 44.

2. Run the %INDNZ_PUBLISH_JAZLIB macro. For more information, see "%INDNZ_PUBLISH_JAZLIB Macro Syntax" on page 45.

%INDNZPJ Macro

The %INDNZPJ macro searches the autocall library for the indnzpj.sas file. The indnzpj.sas file needs to be called before calling the %INDNZ_PUBLISH_JAZLIB macro. The indnzpj.sas file should be in one of the directories listed in the SASAUTOS= system option in your configuration file. If the indnzpj.sas file is not present, the %INDNZPJ macro call (%INDNZPJ; statement) issues the following message:

```
macro indnzpj not defined
```

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: You can 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.

%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 and be 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 42.

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: The %INDNZ_PUBLISH_COMPILEUDF macro is needed only if you plan to use scoring functions to run the scoring models.

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 48.

%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 46.

2. Start SAS 9.3 and submit the following commands in the Enhanced Editor or Program Editor.

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

For more information, see "%INDNZPC Macro" on page 47 and "INDCONN Macro Variable" on page 47.

3. Run the %INDNZ_PUBLISH_COMPILEUDF macro. For more information, see "%INDNZ_PUBLISH_COMPILEUDF Macro Syntax" on page 48.

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

%INDNZPC Macro

The %INDNZPC macro is an autocall library that initializes the %INDNZ_PUBLISH_COMPILEUDF macro.

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: You can 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

Interaction: 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 the "Documentation for Publishing SAS Formats and Scoring Models in Netezza" on page 50.

%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 "%INDNZ_PUBLISH_COMPILEUDF Macro Run Process" on page 46.

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 two 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 9.3 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`

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 8, "Configurations for SAS Model Manager," on page 63.

Documentation for Publishing SAS Formats and Scoring Models 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 6

Administrator's Guide for Oracle

In-Database Deployment Package for Oracle **51**
Prerequisites 51
Overview of the In-Database Package for Oracle 51
Oracle Installation and Configuration Steps 52
Upgrading from or Reinstalling a Previous Version 52
Installing the In-Database Deployment Package for Oracle 52
Creating Users and Objects for the SAS Embedded Process 53
Oracle Permissions 54
Documentation for Scoring Models in Oracle 55

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.

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 9.31).

The in-database deployment package for Oracle must be installed and configured before you can use the %INDOR_PUBLISH_MODEL scoring publishing macro to create scoring files inside the database.

The scoring publishing macros are included in the SAS/ACCESS Interface to Oracle. 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 is installed on your Oracle system 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 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 52 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 52.
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 53.

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) announce to 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.

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-9.31-*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-install-directory*/SASTKInDatabaseServer/9.31/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, move the tkindbsrv-9.31-*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-9.31-*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 in 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-9.31-*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-9.31-n_lax.sh
   ```

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

   ```
   ./tkindbsrv-9.31-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/9.31/bin

   /*path_to_sh_file*/SAS/SASTKInDatabaseServerForOracle/9.31/misc

   /*path_to_sh_file*/SAS/SASTKInDatabaseServerForOracle/9.31/sasexe

   /*path_to_sh_file*/SAS/SASTKInDatabaseServerForOracle/9.31/utilities

   /*path_to_sh_file*/SAS/SASTKInDatabaseServerForOracle/9.31/admin

   /*path_to_sh_file*/SAS/SASTKInDatabaseServerForOracle/9.31/logs

4. For non-shared Oracle home systems, update the contents of the $ORACLE_HOME/hs/admin/extproc.ora file on each node. For 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/9.31/
      SET TKPATH=/path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.31/sasexe
      ```

   Note: Ask your DBA if the ORACLE_HOME environment variable is not set.
5. For non-shared Oracle home systems, update the contents of the $ORACLE_HOME/network/admin/sqlnet.ora file on each node. For 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, 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. The steps to create users and objects is not required for an upgrade or reinstall, unless you want to create and grant privileges for additional users.

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

1. To create the user accounts for Oracle, the DBA must perform the following steps:

   a. Connect as SYS, using the following command:

   ```
   sqlplus sys/<password> as sysdba
   ```

   b. To create and grant user privileges for the SASADMIN user, submit the following statements:

   ```
   create user SASADMIN identified by sasadmin;

   grant connect, resource to SASADMIN;
   grant create table to SASADMIN;
   grant create view to SASADMIN;
   grant create library to SASADMIN;
   grant create any directory to SASADMIN;
   grant drop any directory to SASADMIN;
   grant create public synonym to SASADMIN;
   grant drop public synonym to SASADMIN;
   grant create any context to SASADMIN
   ```

   c. To create and grant user privileges for other users, submit the following SQL statements:

   ```
   # demo, model and nlsmodel are examples of users
   create user demo identified by demo;
   create user model identified by model;
   create user nlsmodel identified by nlsmodel;

   grant connect to demo;
   grant connect, resource to model;
   grant connect, resource to nlsmodel;
   ```

2. 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 directory to ***/path_to_sh_file/SAS/SASTKInDatabaseServerForOracle/9.31/admin***.

   b. Connect as SASADMIN, using the following command:

   ```
   sqlplus sasadmin/<password>
   ```

   c. Submit the following SQL statements:

   ```
   @create_sasepfunc.sql;
   @create_sasepcontext.sql;
   @create_sasepcfg_table.sql;
   @create_saseplog_table.sql;
   ```

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. 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.

Documentation for Scoring Models 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 7

Administrator's Guide for Teradata

In-Database Deployment Package for Teradata **57**
Prerequisites 57
Overview of the In-Database Deployment Package for Teradata 57
Teradata Installation and Configuration Steps 58
Upgrading from or Reinstalling a Previous Version 58
Installing the SAS Formats Library and the SAS Embedded Process 60
Teradata Permissions 62
Documentation for Publishing SAS Scoring Models and Formats in Teradata 62

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.

Teradata server (database) version 13.00 or higher and client (TTU) version 13.00 are required for in-database products. If you want to use the new SAS Embedded Process to publish your scoring models, you need version 13.10.02.01 or higher and client (TTU) version 13.00 or 13.10.

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 2.1 and SAS Embedded Process 9.31). 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.

The format and scoring publishing macros are included in SAS/ACCESS Interface to Teradata. 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 starting in November 2011, 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 61.

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 58.
2. Install the in-database deployment package.

   For more information, see "Installing the SAS Formats Library and the SAS Embedded Process" on page 60.
3. Install the SAS Embedded Process support functions.

   For more information, see "Installing the SAS Embedded Process Support Functions" on page 61.
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 8, "Configurations for SAS Model Manager," on page 63.

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 60.

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-9.31-1
   ```

   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 60.

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-2.1-1 replacing jazxfbrs-9.2-1.6 or tkindbsrv-9.31-2 replacing tkindbsrv-9.31-1.

   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 shutdown the SAS Embedded Process. For more information, see "Controlling the SAS Embedded Process" on page 61.

   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-9.31-*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-9.31-*version*.

4. (Optional) To confirm removal of the package before installing the new package, run this command on all nodes:

   ```
   psh "rpm -q package-name"
   ```

   package-name is either jazxfbrs.9.*version*, accelterafmt-2.*version*, or tkindbsrv-9.31-*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-2.1-*n*.x86_64.rpm and tkindbsrv-9.31-*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-2.1-*n*.x86_64.rpm file is located in the ***SAS-install-directory*/SASFormatsLibraryforTeradata/2.1/TeradataonLinux/** directory. The tkindbsrv-9.31-*n*.x86_64.rpm file is located in the **SAS-install-directory/SASTKInDatabaseServer/9.31/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. 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 Release 3.05.01B035–5713–011K January 2011*, located at **http://www.info.teradata.com/edownload.cfm?itemid=110550001**. 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 60.
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 a Non-VM&F installation.
5. If the install is successful, *accelterfmt-2.1-n* or *tkindbsrv-9.31-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 install 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 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 on the Teradata Web site. For access to the package, contact your local Teradata account representative.

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.

Command	Action performed
`CALL DBCEXTENSION.SERVER CONTROL ('STATUS', :A);`	Provides the status of the SAS Embedded Process.
`CALL DBCEXTENSION.SERVER CONTROL ('SHUTDOWN', :A);`	Shuts down the SAS Embedded Process. *Note:* You cannot shutdown until all queries are complete.

Command	Action performed
`CALL DBCEXTENSION.SERVER CONTROL ('DISABLE', :A);`	Stops new queries from being started. Queries that are currently running continue to run until they are complete.
`CALL DBCEXTENSION.SERVER CONTROL ('ENABLE', :A);`	Enables new queries to start running.

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:

EXECUTE PROCEDURE ON SAS_SYSFNLIB TO *userid*

EXECUTE FUNCTION ON SAS_SYSFNLIB TO *userid*

EXECUTE FUNCTION ON SAS_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 8, "Configurations for SAS Model Manager," on page 63.

Documentation for Publishing SAS Scoring Models and Formats in Teradata

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 8

Configurations for SAS Model Manager

Preparing a Database for Use with SAS Model Manager 63
Prerequisites 63
Overview of Preparing a Database for Use with SAS Model Manager 63
Configuring a Database 64
Finding the JDBC JAR Files 65

Preparing a Database for Use with SAS Model Manager

Prerequisites

SAS Foundation, the SAS/ACCESS Interface, and the in-database deployment package for the database must be installed and configured before you can prepare a database for use with SAS Model Manager. For more information, see the chapter for your type of database in this guide. Here are the databases that can be used with SAS Model Manager:

- DB2
- Greenplum
- Netezza
- Teradata

Overview of Preparing a Database for Use with SAS Model Manager

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. Additional configuration steps are required to prepare the database for publishing and scoring in SAS Model Manager.

During the installation and configuration of SAS 9.3 products, the SAS Model Manager In-Database Scoring Scripts product is installed on the middle-tier server or another server tier using a 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`**

In the script installation directory, includes a directory that specifies the version of SAS Model Manager, which is currently 3.1. 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

To enable users to publish scoring functions to a database from SAS Model Manager, follow these steps:

1. Create a separate database where the tables can be stored.
2. Set the user access permissions for the database.
   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 27
      - "Greenplum Permissions" on page 39
      - "Netezza Permissions" on page 48
      - "Teradata Permissions" on page 62

   b. GRANT CREATE and DROP permissions for tables so that 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 those tables or views.
3. Navigate to the ***sasinstalldir*\SASModelManagerInDatabaseScoringScripts\3.1\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 Web application servers are the following:

   JBoss
   ***JBoss_Home*\server\SASServer1\lib**

   An example of the directory path is the following: **C:\JBoss4.3.0.GA \server\SASServer1\lib**

   WebLogic
   ***sasconfigdir*\Lev#\Web\SASDomain\lib**

   An example of the directory path is the following: **C:\SAS\Config \Lev1\Web\SASDomain\lib**

   WebSphere
   ***WebSphere_HOME*\lib**

   An example of the directory path is the following: **C:\Program Files\IBM \WebSphere7\AppServer\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 65.

5. Restart the SAS servers on the Web application server.
   - JBoss: Use JBoss services or commands to restart the SAS servers.
   - WebLogic: Use the WebLogic Administration Console or commands to restart the SAS servers.
   - WebSphere: Use the WebSphere Admin Console or commands to restart the SAS servers.

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 http://jdbc.postgresql.org/download.html. An example of a JDBC driver name is **postgresql-9.1-901.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 Web site at http://

www.teradata.com. Select **Support & Downloads** ⇨ **Downloads** ⇨ **Teradata JDBC Driver**.

For more information about the database versions that are supported, see the *SAS Scoring Accelerator System Requirements* at **http://www.sas.com/technologies/analytics/datamining/scoring_acceleration/#section=5**.

Index

Special Characters

%INDAC_PUBLISH_FORMATS macro 3
%INDAC_PUBLISH_MODEL macro 3
%INDB2_PUBLISH_COMPILEUDF macro 20
  running 21
  syntax 22
%INDB2_PUBLISH_DELETEUDF macro 24
  running 24
  syntax 26
%INDB2_PUBLISH_FORMATS macro 9
%INDB2_PUBLISH_MODEL macro 9
%INDB2PC macro 21
%INDB2PD macro 25
%INDGP_PUBLISH_COMPILEUDF macro 35
  running 35
  syntax 37
%INDGP_PUBLISH_FORMATS macro 31
%INDGP_PUBLISH_MODEL macro 31
%INDGPPC macro 36
%INDNZ_PUBLISH_COMPILEUDF macro 46
  running 46
  syntax 48
%INDNZ_PUBLISH_FORMATS macro 41
%INDNZ_PUBLISH_JAZLIB macro 44
  running 44
  syntax 45
%INDNZ_PUBLISH_MODEL macro 41
%INDNZPC macro 47
%INDNZPJ macro 44
%INDOR_PUBLISH_MODEL macro 51
%INDTD_PUBLISH_FORMATS macro 57
%INDTD_PUBLISH_MODEL macro 57

A

ACTION= argument
  %INDB2_PUBLISH_COMPILEUDF macro 23
  %INDB2_PUBLISH_DELETEUDF macro 26
  %INDGP_PUBLISH_COMPILEUDF macro 38
  %INDNZ_PUBLISH_COMPILEUDF macro 48
  %INDNZ_PUBLISH_JAZLIB macro 45
Aster nCluster
  documentation for publishing formats and scoring models 7
  in-database deployment package 3
  installation and configuration 4
  permissions 6
  SAS Embedded Process 3
  SAS/ACCESS Interface 3
  SQL/MR functions 4

B

binary files
  for Aster nCluster 4
  for DB2 functions 15
  for Greenplum functions 33
  for Netezza functions 43

C

COMPILER_PATH= argument
  %INDB2_PUBLISH_COMPILEUDF macro 23
configuration
  Aster nCluster 3
  DB2 10
  Greenplum 32
  Netezza 42
  Oracle 51

Teradata 58

D

DATABASE= argument
%INDB2_PUBLISH_COMPILEUDF macro 23
%INDB2_PUBLISH_DELETEUDF macro 26
%INDGP_PUBLISH_COMPILEUDF macro 37
%INDNZ_PUBLISH_COMPILEUDF macro 48
%INDNZ_PUBLISH_JAZLIB macro 45
DB2
documentation for publishing formats or scoring models 29
function publishing process 10
in-database deployment package 9
installation and configuration 10
JDBC Driver 65
permissions 27
preparing for SAS Model Manager use 63
SAS/ACCESS Interface 9
unpacking self-extracting archive files 15, 16
DB2IDA command 19
DB2PATH= argument
%INDB2_PUBLISH_COMPILEUDF macro 22
DB2SET command 16
DB2SET command syntax for DB2 18
documentation
for publishing formats and scoring models in Aster nCluster 7
for publishing formats and scoring models in DB2 29
for publishing formats and scoring models in Greenplum 39
for publishing formats and scoring models in Netezza 50
for publishing formats and scoring models in Teradata 62
for publishing scoring models in Oracle 55

F

formats library
DB2 installation 14
Greenplum installation 33
Netezza installation 42, 43
Teradata installation 60
function publishing process
DB2 10
Greenplum 32
Netezza 42
functions
SAS_COMPILEUDF (DB2) 14, 20, 26
SAS_COMPILEUDF (Greenplum) 33, 35, 38
SAS_COMPILEUDF (Netezza) 43, 46
SAS_COPYUDF (Greenplum) 38
SAS_DEHEXUDF (Greenplum) 38
SAS_DELETEUDF (DB2) 14, 24, 26
SAS_DIRECTORYUDF (Greenplum) 38
SAS_DIRECTORYUDF (Netezza) 46
SAS_HEXTOTEXTUDF (Netezza) 46
SAS_PUT( ) (Aster nCluster) 3
SAS_PUT( ) (DB2) 9
SAS_PUT( ) (Greenplum) 31
SAS_PUT( ) (Netezza) 42
SAS_PUT( ) (Teradata) 57
SAS_SCORE( ) (Aster nCluster) 4
SQL/MR (Aster nCluster) 4

G

global variables
See variables
Greenplum
documentation for publishing formats and scoring models 39
function publishing process 32
in-database deployment package 31
installation and configuration 32
JDBC Driver 65
permissions 39
preparing for SAS Model Manager use 63
SAS/ACCESS Interface 31
unpacking self-extracting archive files 33

I

in-database deployment package for Aster nCluster
overview 3
prerequisites 3
in-database deployment package for DB2
overview 9
prerequisites 9
in-database deployment package for Greenplum
overview 31
prerequisites 31
in-database deployment package for Netezza

overview 41
prerequisites 41
in-database deployment package for Oracle
overview 51
prerequisites 51
in-database deployment package for Teradata
overview 57
prerequisites 57
INDCONN macro variable 21, 25, 36, 44, 47
installation
Aster nCluster 3
DB2 10
Greenplum 32
Netezza 42
Oracle 51
SAS Embedded Process (Aster nCluster) 3
SAS Embedded Process (DB2) 10, 14
SAS Embedded Process (Oracle) 51
SAS Embedded Process (Teradata) 58
SAS formats library 14, 33, 43, 60
Teradata 58

J

JDBC Driver
DB2 65
Greenplum 65
Netezza 65
Teradata 65

M

macro variables
See variables
macros
%INDAC_PUBLISH_FORMATS 3
%INDAC_PUBLISH_MODEL 3
%INDB2_PUBLISH_COMPILEUDF 20, 22
%INDB2_PUBLISH_DELETEUDF 24, 26
%INDB2_PUBLISH_FORMATS 9
%INDB2_PUBLISH_MODEL 9
%INDB2PC 21
%INDB2PD 25
%INDGP_PUBLISH_COMPILEUDF 35, 37
%INDGP_PUBLISH_FORMATS 31
%INDGP_PUBLISH_MODEL 31
%INDGPPC 36
%INDNZ_PUBLISH_COMPILEUDF 46, 48
%INDNZ_PUBLISH_FORMATS 41
%INDNZ_PUBLISH_JAZLIB 44, 45
%INDNZ_PUBLISH_MODEL 41
%INDNZPC 47
%INDNZPJ 44
%INDOR_PUBLISH_MODEL 51
%INDTD_PUBLISH_FORMATS 57
%INDTD_PUBLISH_MODEL 57
Model Manager configuration 63

N

Netezza
documentation for publishing formats and scoring models 50
function publishing process 42
in-database deployment package 41
installation and configuration 42
JDBC Driver 65
permissions 48
preparing for SAS Model Manager use 63
publishing SAS formats library 44
SAS/ACCESS Interface 41
unpacking self-extracting archive files 43

O

OBJNAME= argument
%INDB2_PUBLISH_COMPILEUDF macro 23
OBJPATH= argument
%INDGP_PUBLISH_COMPILEUDF macro 37
Oracle
documentation for publishing formats and scoring models 55
in-database deployment package 51
permissions 54
SAS Embedded Process 51
SAS/ACCESS Interface 51
OUTDIR= argument
%INDB2_PUBLISH_COMPILEUDF macro 23
%INDB2_PUBLISH_DELETEUDF macro 26
%INDGP_PUBLISH_COMPILEUDF macro 38
%INDNZ_PUBLISH_COMPILEUDF macro 48
%INDNZ_PUBLISH_JAZLIB macro 46

P

permissions
  for Aster nCluster 6
  for DB2 27
  for Greenplum 39
  for Netezza 48
  for Oracle 54
  for Teradata 62
PSFTP (DB2) 10
publishing
  Aster nCluster permissions 6
  DB2 permissions 27
  functions in DB2 10
  functions in Greenplum 32
  functions in Netezza 42
  Greenplum permissions 39
  Netezza permissions 48
  Oracle permissions 54
  Teradata permissions 62

R

reinstalling a previous version
  Aster nCluster 4
  DB2 11
  Greenplum 33
  Netezza 43
  Oracle 52
  Teradata 58
RPM file (Teradata) 60

S

SAS_COMPILEUDF function
  actions for DB2 20
  actions for Greenplum 35
  actions for Netezza 46
  binary files for DB2 14
  binary files for Greenplum 33
  binary files for Netezza 43
  validating publication for DB2 26
  validating publication for Greenplum 38
SAS_COPYUDF function 35
  validating publication for Greenplum 38
SAS_DEHEXUDF function 35
  validating publication for Greenplum 38
SAS_DELETEUDF function
  actions for DB2 24
  binary files for DB2 14
  validating publication for DB2 26
SAS_DIRECTORYUDF function 35, 46
  validating publication for Greenplum 38
SAS_HEXTOTEXTUDF function 46
SAS_PUT( ) function
  Aster nCluster 3
  DB2 10
  Greenplum 31
  Netezza 42
  Teradata 57
SAS_SCORE( ) function
  publishing 4
  validating publication for Aster nCluster 6
SAS_SYSFNLIB (Teradata) 61
SAS Embedded Process
  Aster nCluster 3
  check status (DB2) 19
  check status (Teradata) 61
  disable or enable (DB2) 19
  disable or enable (Teradata) 61
  Oracle 51
  shutdown (DB2) 19
  shutdown (Teradata) 61
  Teradata 57
  upgrading from a previous version (Aster nCluster) 4
  upgrading from a previous version (Oracle) 52
  upgrading from a previous version (Teradata) 58
SAS Embedded Process support functions (Teradata) 61
SAS FILENAME SFTP statement (DB2) 10
SAS formats library
  DB2 14
  Greenplum 33
  Netezza 43, 44
  Teradata 60
  upgrading from a previous versions (DB2) 11
  upgrading from a previous versions (Netezza) 43
  upgrading from a previous versions (Teradata) 58
SAS Formats Library
  upgrading from a previous version (Greenplum) 33
SAS Foundation 3, 9, 31, 41, 51, 57
SAS In-Database products 1
SAS/ACCESS Interface to Aster nCluster 3
SAS/ACCESS Interface to DB2 9
SAS/ACCESS Interface to Greenplum 31
SAS/ACCESS Interface to Netezza 41
SAS/ACCESS Interface to Oracle 51
SAS/ACCESS Interface to Teradata 57
SASLIB database (Netezza) 46

SASLIB schema
DB2 21, 24
Greenplum 35
SASUDF_COMPILER_PATH global variable 20
SASUDF_DB2PATH global variable 20
scoring functions in SAS Model Manager 63
self-extracting archive files
unpacking for Aster nCluster 4
unpacking for DB2 15, 16
unpacking for Greenplum 33
unpacking for Netezza 43
SFTP statement 10
SQL/MR functions (Aster nCluster) 4
SSH software (DB2) 10

T

tables
creating for SAS Model Manager 64
Teradata
documentation for publishing formats and scoring models 62
in-database deployment package 57
installation and configuration 58
JDBC Driver 65
permissions 62
preparing for SAS Model Manager use 63
SAS Embedded Process 57
SAS Embedded Process support functions 61
SAS/ACCESS Interface 57
Teradata Parallel Upgrade Tool 60

U

unpacking self-extracting archive files
for Aster nCluster 4
for DB2 15, 16
for Greenplum 33
for Netezza 43
upgrading from a previous version
Aster nCluster 4
DB2 11
Greenplum 33
Netezza 43
Oracle 52
Teradata 58

V

validating publication of functions and variables for DB2 26
validating publication of functions for Aster nCluster 6
validating publication of functions for Greenplum 38
variables
INDCONN macro variable 21, 25, 36, 44, 47
SASUDF_COMPILER_PATH global variable 20
SASUDF_DB2PATH global variable 20