THE POWER TO KNOW.

SAS® Scoring Accelerator 1.7 for Teradata User's Guide

SAS® Documentation

Contents

What's New in SAS Scoring Accelerator 1.7 for Teradata

Overview

The SAS Scoring Accelerator 1.7 for Teradata has the following new features.

- availability of the Score Code Export node software in SAS Enterprise Miner

Score Code Export Node Software Availability

Starting with version 1.7, the Score Code Export node software is available in SAS Enterprise Miner. In previous versions of the SAS Scoring Accelerator for Teradata, the software was provided in a ZIP file.

Chapter 1
Introduction to the SAS Scoring Accelerator for Teradata

Overview of the SAS Scoring Accelerator for Teradata

When using conventional processing to access data inside a Teradata Enterprise Data Warehouse (EDW), SAS Enterprise Miner asks the SAS/ACCESS engine for all rows of the table being processed. The SAS/ACCESS engine generates an SQL SELECT * statement that is passed to the Teradata EDW. That SELECT statement fetches all the rows in the table, and the SAS/ACCESS engine returns them to SAS Enterprise Miner. As the number of rows in the table grows over time, network latency grows because the amount of data that is fetched from the Teradata EDW to the SAS scoring process increases.

The SAS Scoring Accelerator for Teradata embeds the robustness of SAS Enterprise Miner scoring models directly in the highly scalable Teradata database. By using the SAS In-Database technology and the SAS Scoring Accelerator for Teradata, the scoring process is done inside the data warehouse, and thus does not require the transfer of data.

The SAS Scoring Accelerator for Teradata takes the models that are developed by SAS Enterprise Miner and translates them into scoring functions that can be deployed inside Teradata. After the scoring functions are published, the functions extend the Teradata SQL language and can be used in SQL statements like other Teradata functions.

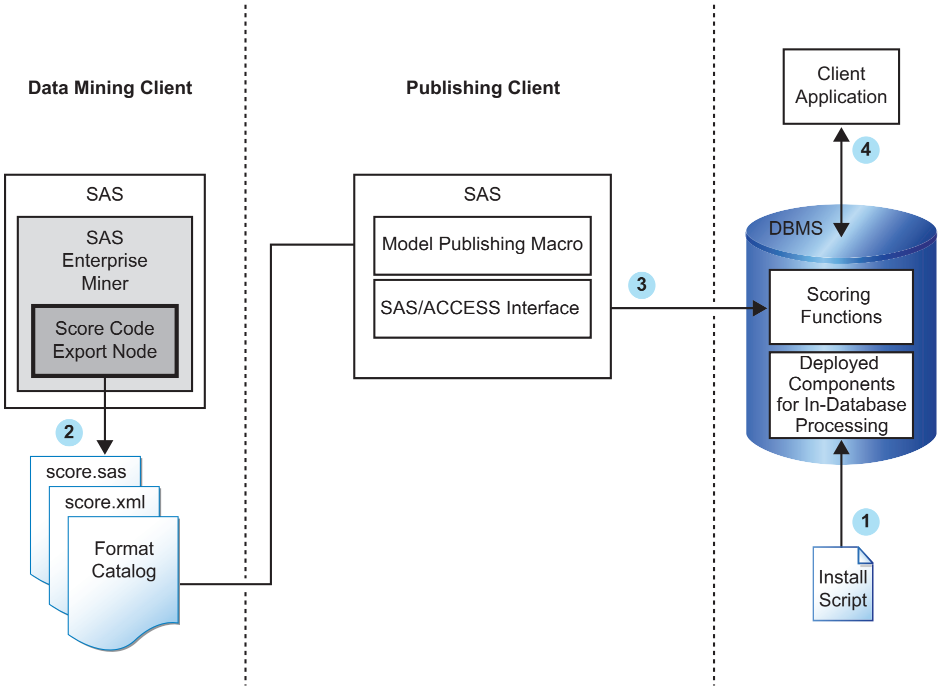
The SAS Scoring Accelerator for Teradata consists of two components:

- the Score Code Export node in SAS Enterprise Miner. This extension exports the model scoring logic, including metadata about the required input and output variables, from SAS Enterprise Miner.
- the publishing client that includes the %INDTD_PUBLISH_MODEL macro. This macro translates the scoring model into .c and .h files for creating the scoring functions and generates a script of Teradata commands for registering the scoring functions. The publishing client then uses the SAS/ACCESS Interface to Teradata to publish the scoring functions to Teradata.

How the SAS Scoring Accelerator for Teradata Works

Using SAS Enterprise Miner, you can generate SAS DATA step code that contains scoring functions. The SAS Scoring Accelerator for Teradata takes the scoring model code, the associated property file that contains model inputs and outputs, and a catalog of user-defined formats, and deploys, or publishes, them to the Teradata EDW. Inside the Teradata EDW, one or more scoring functions are created and registered for use in SQL queries. Figure 1.1 illustrates this process.

Figure 1.1 *Process Flow Diagram*



1 Install the components that are necessary for in-database processing in the Teradata EDW.

For more information, see "Overview of Deployed Components for In-Database Processing" on page 5.

Note: This is a one-time installation process.

2 Use SAS Enterprise Miner to create a scoring model, and use the Score Code Export node to export files that are used to create the scoring functions to a score output directory.

For more information, see Chapter 3, "Exporting the Scoring Model Files from SAS Enterprise Miner," on page 7.

3 Start SAS 9.2 and run the SAS publishing macros, which create the files that are needed to build the scoring functions and publish those files to the Teradata EDW.

For more information, see Chapter 4, "Publishing the Scoring Model Files," on page 19.

4 After the scoring functions are created, they are available to use in any SQL expression in the same way that Teradata built-in functions are used.

For more information, see Chapter 5, “Scoring Functions Inside the Teradata EDW,” on page 27.

Chapter 2

Deployed Components for In–Database Processing

Overview of Deployed Components for In-Database Processing

The component that is deployed is the SAS 9.2 Formats Library for Teradata. The SAS 9.2 Formats Library for Teradata contains many of the formats that are available in Base SAS and processes any formats that might be included in your scoring model.

Components that are deployed to Teradata for in-database processing are contained in an RPM installation file for Teradata nodes running Linux. The RPM file can be found on the SAS Software Depot.

Note: This library is also used by the %INDTD_PUBLISH_FORMATS macro and the SAS_PUT( ) function. For more information about these features, see "Deploying and Using SAS Formats in Teradata" in *SAS/ACCESS 9.2 for Relational Databases: Reference*.

For more information about creating the SAS Software Depot, see the instructions in your Software Order e-mail. For more information about installing and configuring these components, see the *SAS In-Database Products: Administrator's Guide*.

Chapter 3

Exporting the Scoring Model Files from SAS Enterprise Miner

Overview of the Score Code Export Node

Users of SAS Enterprise Miner develop data mining models that use measured attributes to either characterize or predict the value of an event. These models are developed on historical data where an event has been measured or inferred. The models are then applied to new data for which the attributes are known, but the event has not yet occurred. For example, a model can be created based on a credit institution’s records of payments that customers made and missed last year and then used to predict which customers will miss payments this year.

SAS Enterprise Miner creates SAS language score code for the purpose of scoring new data. Users run this code in production systems to make business decisions for each record of new data.

The Score Code Export node is an extension for SAS Enterprise Miner that exports files that are necessary for score code deployment. Extensions are programmable add-ins for the SAS Enterprise Miner environment.

The following icon is the Score Code Export node as it appears in a SAS Enterprise Miner process flow diagram.

The following files are exported by the Score Code Export node:

- the SAS scoring model program (score.sas).
- a properties file that contains a description of the variables that are used and created by the scoring code (score.xml).
- a format catalog, if the scoring program contains user-defined formats.
- an XML file containing descriptions of the final variables that are created by the scoring code. This file can be kept for decision-making processes.
- a ten-row sample of the scored data set showing typical cases of the input attributes, intermediate variables, and final output variables. This data set can be used to test and debug new scoring processes.
- a ten-row sample table of the training data set showing the typical cases of the input attributes used to develop the score code.

For more information about the exported files, see "Output Files" on page 11. For more information about using SAS Enterprise Miner, see the SAS Enterprise Miner Help.

Using the Score Code Export Node Compared with Registering Models on the SAS Metadata Server

SAS Enterprise Miner can register models directly in the SAS Metadata Server. Models registered in the SAS Metadata Server are used by SAS Data Integration Studio, SAS Enterprise Guide, and SAS Model Manager for creating, managing, and monitoring production and analytical scoring processes.

The Score Code Export node exports score code created by SAS Enterprise Miner into a format that can be used by the SAS Scoring Accelerator for Teradata. The exported files are stored in a directory, not the SAS Metadata Server.

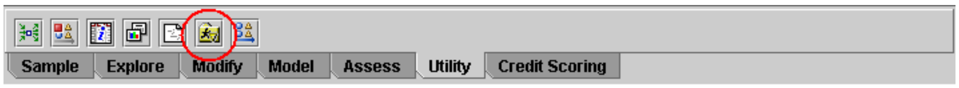
The Score Code Export node does not replace the functionality of registering models in the SAS Metadata Server.

Using the Score Code Export Node

Using the Score Code Export Node in a Process Flow Diagram

The **Score Code Export node** icon is located on the Utility tab, as shown in Figure 3.1:

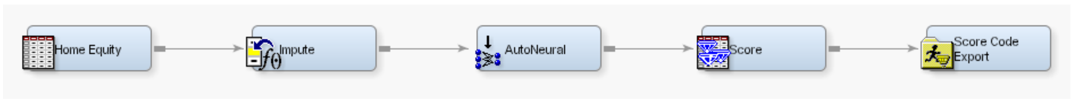
Figure 3.1 *The Diagram Toolbar with the SAS Score Code Export Node Icon Highlighted*



To use the Score Code Export node, you need a process flow diagram that contains nodes that produce score code and that flow to a Score node. The Score node aggregates the score code for the entire analysis path. The Score node must precede the Score Code Export node in the process flow diagram.

Figure 3.2 shows a valid data mining process for exporting score code:

Figure 3.2 *Data Mining Process Flow Diagram*



Requirement: The Score Code Export node exports score code that contains only one DATA step. For a list of SAS Enterprise Miner nodes that produce score code, see "SAS Enterprise Miner Tools Production of Score Code" on page 14.

After the process flow diagram is in place, set the properties for the Score node and the Score Code Export node:

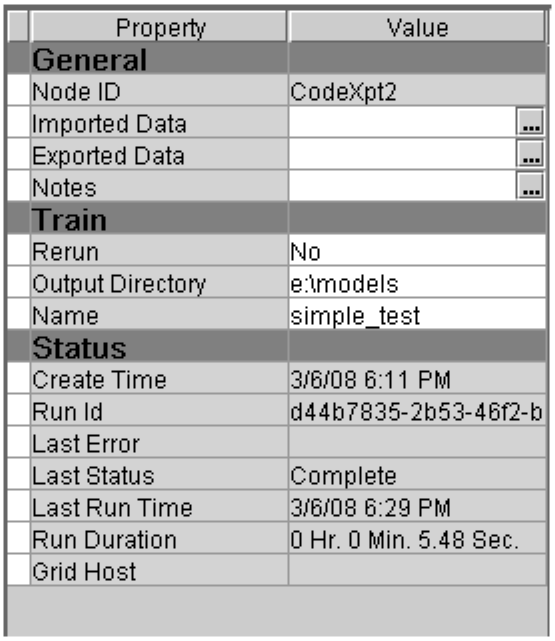
1. Select the Score node. Ensure that the following properties are set to their default value of Yes:
   - **Use Output Fixed Names**
   - **C Score**
2. Select the Score Code Export node and set the properties. The **Output Directory** property specifies the directory to store the export files. The **Name** property specifies the folder that contains the output files created by the Score Code Export node. For information about the properties, see "Score Code Export Node Properties" on page 9.

After the properties are set, you are ready to export the score code. Right-click the Score Code Export node and select **Run**. When SAS Enterprise Miner completes processing, the Run Status window opens to indicate that the run completed. Click the **Results** button to view the output variables and the listing output. For information about the output, see "Output Created by the Score Code Export Node" on page 10.

Score Code Export Node Properties

When the Score Code Export node is selected in the diagram workspace, the Properties panel displays all of the properties that the node uses and their associated values, as shown in Figure 3.3.

Figure 3.3 Properties Panel



Property	Value
General	
Node ID	CodeXpt2
Imported Data	...
Exported Data	...
Notes	...
Train	
Rerun	No
Output Directory	e:\models
Name	simple_test
Status	
Create Time	3/6/08 6:11 PM
Run Id	d44b7835-2b53-46f2-b
Last Error	
Last Status	Complete
Last Run Time	3/6/08 6:29 PM
Run Duration	0 Hr. 0 Min. 5.48 Sec.
Grid Host	

The following Train properties are associated with the Score Code Export node:

- **Rerun** – Use this property to force the node to run again. This property is useful if the macro variable controlling the target directory and folder name has changed.
- **Output Directory** – Enter a fully qualified name for the location of an output directory to contain the score code files. If no directory is entered, a default directory named Score is created in the SAS Enterprise Miner project directory. You can change the value of the default directory by setting the &EM_SCOREDIR=*directory* macro variable in the SAS Enterprise Miner project start-up code or server start-up code.
- **Name** – Enter the name of the model that you are creating. The name is used to create a new subdirectory in the output directory that contains the exported score files. If no name is entered, a default name is generated as a combination of the &SYSUSERID automatic macro variable and an incremental index (for example, userID, userID_2, userID_3).

  You can replace the &SYSUSERID automatic macro variable with a custom name by setting the &EM_SCOREFOLDER=*score-folder-name* macro variable in the SAS Enterprise Miner project start-up code or server start-up code. An incremental index preceded by an underscore is added to *score-folder-name*.

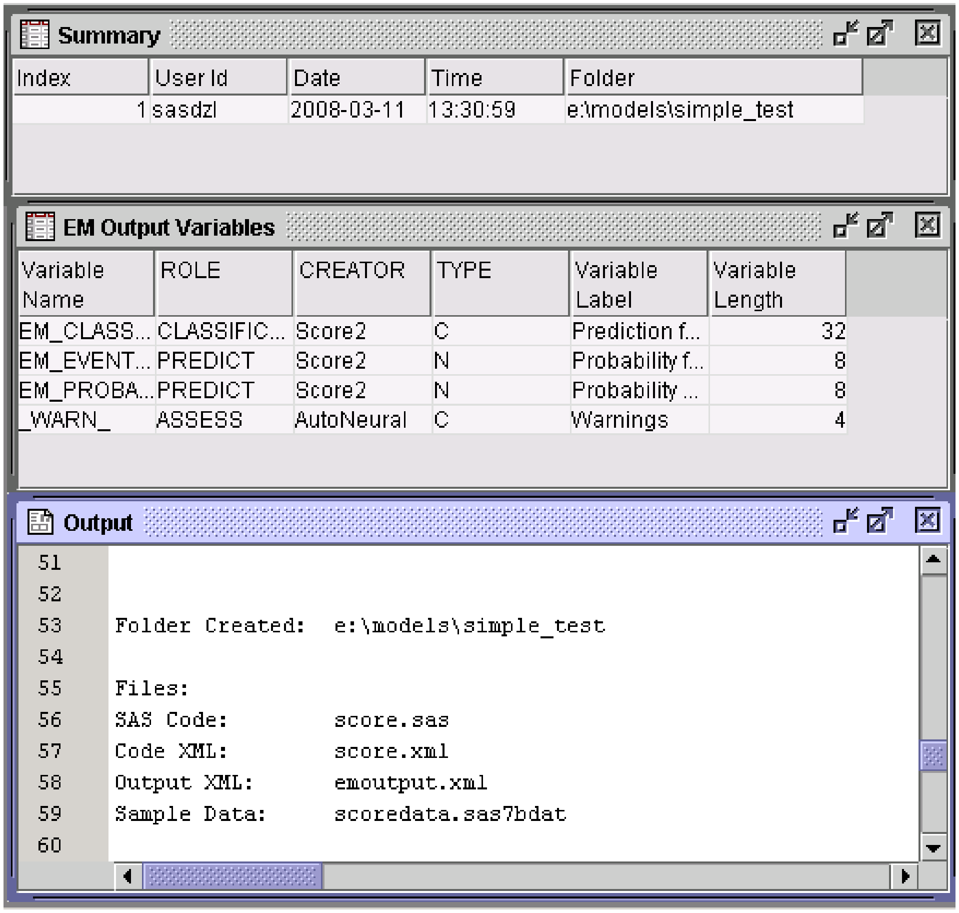
The General and Status properties for the Score Code Export node function just as they do for other nodes.

Output Created by the Score Code Export Node

Results Window

Using the values set in the Properties panel (Figure 3.3), the Score Code Export node creates the following output in the Results window:

Figure 3.4 *Results Using Sample Properties*



Summary

Index	User Id	Date	Time	Folder
1	sasdzl	2008-03-11	13:30:59	e:\models\simple_test

EM Output Variables

Variable Name	ROLE	CREATOR	TYPE	Variable Label	Variable Length
EM_CLASS...	CLASSIFIC...	Score2	C	Prediction f...	32
EM_EVENT...	PREDICT	Score2	N	Probability f...	8
EM_PROBA...	PREDICT	Score2	N	Probability ...	8
WARN	ASSESS	AutoNeural	C	Warnings	4

Output

```


Folder Created:  e:\models\simple_test

Files:
SAS Code:        score.sas
Code XML:        score.xml
Output XML:      emoutput.xml
Sample Data:     scoredata.sas7bdat

```

Output Files

The Score Code Export node writes the following output files, and a format catalog, if applicable, to the location specified by the Output Directory property. These files are used as input to the %INDTD_PUBLISH_MODEL macro that creates the scoring functions.

File or Folder	Description
score.sas	SAS language score code created by SAS Enterprise Miner. This code can be used directly in a SAS program. A sample program based on the properties shown in Figure 3.3 looks like this: `data testout ;` `set simpletest.scoredata ;` `%include “c:\models\simpletest\score.sas”;` `run;`
score.xml	A description of the variables that are used and created by the scoring code. XML files are created by a machine process for the use of machine processes. Do not edit the XML file. **Restriction:** The maximum number of input variables for a scoring function is 128.

File or Folder	Description
emoutput.xml	A description of the final variables that are created by the scoring code. This file can be kept for decision-making processes. These variables include the primary classification, prediction, probability, segment, profit, and loss variables created by a data mining process. The list does not include intermediate variables created by the analysis. For more information about these variables, see "Fixed Variable Names" on page 13. *Note:* The emoutput.xml file is not used by the %INDTD_PUBLISH_MODEL macro.
scoredata.sas7bdat	A ten-row sample of the scored data set showing typical cases of the input attributes, intermediate variables, and final output variables. Use this data set to test and debug new scoring processes. *Note:* The scoredata.sas7bdat file is not used by the %INDTD_PUBLISH_MODEL macro.
traindata.sas7bdat	A ten-row sample table of the training data set showing typical cases of the input attributes used to develop the score code. *Note:* The traindata.sas7bdat file is not used by the %INDTD_PUBLISH_MODEL macro.
Format Catalog	If the training data contains SAS user-defined formats, the Score Code Export node creates a format catalog. The catalog contains the user-defined formats in the form of a look-up table. This file has an extension of .sas7bcat.

Output Variables

The score code produced by SAS Enterprise Miner creates both intermediate variables, such as imputed values of missing values, transformations, and encodings; and output variables, such as predicted value and probability. Any of these created variables can be used in a scoring process.

TIP The number of input parameters on a scoring function has a direct impact on performance. The more parameters there are, the more time it takes to score a row. A recommended best practice is to make sure that only variables that are involved in a model score evaluation are exported from SAS Enterprise Miner.

The most important output variables for the scoring process follow a naming convention using a prefix, as shown in the following table.

Role	Type	Prefix	Key	Suffix	Example
Prediction	N	P_	Target variable name		P_amount
Probability	N	P_	Target variable name	Predicted event value	P_purchaseYES P_purchaseNO

Role	Type	Prefix	Key	Suffix	Example
Classification	$	I_	Target variable name		I_purchase
Expected Profit	N	EP_	Target variable name		EP_conversion
Expected Loss	N	EL_	Target variable name		EL_conversion
Return on Investment	N	ROI_	Target variable name		ROI_conversion
Decision	$	D_	Target variable name		D_conversion
Decision Tree Leaf	N	_NODE_			_NODE_
Cluster number or SOM cell ID	N	_SEGMENT_			_SEGMENT_

Fixed Variable Names

The Score node of SAS Enterprise Miner maps the output variable names to fixed variable names. This mapping is appropriate in cases where there is only one prediction target or one classification target. In other cases, refer to the output variable names described in the previous table.

Using the fixed variable names enables scoring users to build processes that can be reused for different models without changing the code that processes the outputs. These fixed names are listed in the emoutput.xml file and are described in the following table. Most scoring processes return one or more of these variables.

Role	Type	Fixed Name	Description
Prediction	N	EM_PREDICTION	The prediction value for an interval target.
Probability	N	EM_PROBABILITY	The probability of the predicted classification, which can be any one of the target variable values.

Role	Type	Fixed Name	Description
Probability	N	EM_ EVENTPROBABILITY	The probability of the target event. By default this is the first value in descending order. This is often the event of interest. The user can control the ordering in SAS Enterprise Miner.
Classification	$	EM_ CLASSIFICATION	The predicted target class value.
Expected Profit	N	EM_PROFIT	Based on the selected decision.
Expected Loss	N	EM_LOSS	Based on the selected decision.
Return on Investment	N	EM_ROI	Based on the selected decision.
Decision	$	EM_DECISION	Optimal decision based on a function of probability, cost, and profit or loss weights.
Decision Tree Leaf, Cluster number, or SOM cell ID	N	EM_SEGMENT	Analytical customer segmentation.

SAS Enterprise Miner Tools Production of Score Code

The following table shows the types of score code created by each node in SAS Enterprise Miner. Users can develop their own nodes, known as extension nodes, which can create either SAS DATA step or SAS program score code. However, this code is not converted to PMML, C, or Java.

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata
Sample						
Input Data	*	*	*	*	*	*
Sample	*	*	*	*	*	*
Partition	*	*	*	*	*	*
Append	N	Y	N	N	N	N
Merge	N	Y	N	N	N	N
Time Series	N	Y	N	N	N	N

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata
Filter	Y When the user keeps the created filter variable.	*	N	Y	Y	Y
Explore						
Association	N	Y	Y	N	N	N
Cluster	Y	N	Y	Y	Y	Y
DMDB	*	*	*	*	*	*
Graph Explore	*	*	*	*	*	*
Market Basket	N	Y	N	N	N	N
Multiplot	*	*	*	*	*	*
Path	N	Y	Y	N	N	N
SOM	Y	N	N	Y	Y	Y
Stat Explore	*	*	*	*	*	*
Text Miner	N	Y	N	N	N	N
Variable Clustering	Y	N	N	Y	Y	Y
Variable Selection	Y	N	N	Y	Y	Y
Drop	*	*	*	*	*	*
Impute	Y	N	Y	Y	Y	Y
Interactive Binning	Y	N	N	Y	Y	Y
Replacement	Y	N	N	Y	Y	Y
Principle Components	Y	N	N	Y	Y	Y
Rules Builder	Y	N	N	Y	Y	Y
Transform Variables	Y	N	N	Y	Y	Y
Model						

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata
Autoneural	Y	N	Y	Y	Y	Y
Decision Tree	Y	N	Y	Y	Y	Y
Dmine Regression	Y	N	Y	Y	Y	Y
Dmine Neural	Y	N	N	Y	Y	Y
Ensemble	Y	N	N	Y	Y	Y
Gradient Boosting	Y	N	N	Y	Y	Y
MBR	N	Y	N	N	N	N
Model Import	*	*	*	*	*	*
Neural Network	Y	N	Y	Y	Y	Y
Partial Least Squares	Y	N	N	Y	Y	Y
Rule Induction	Y	N	N	Y	Y	Y
SVM — Linear Kernel	Y	N	Y	Y	Y	Y
SVM — Nonlinear Kernel	N	Y	N	N	N	N
Two Stage	Y	N	N	Y	Y	Y
Assess						
Cutoff	Y	N	N	Y	Y	Y
Decisions	Y	N	N	Y	Y	Y
Model Comparison	Y	N	N	Y	Y	Y
Score	Y	N	N	Y	Y	Y
Segment Profile	*	*	*	*	*	*
Utility						
Control Point	*	*	*	*	*	*

Node	**SAS DATA Step**	**SAS Program**	**PMML**	**C**	**Java**	**Teradata**
Start Groups	Y	N	N	Y	Y	Y
End Groups	Y	N	N	Y	Y	Y
Metadata	*	*	*	*	*	*
Reporter	*	*	*	*	*	*
SAS Code The user can enter either SAS DATA step code or SAS program code	Y	Y	N	N	N	N
Credit Scoring						
Credit Exchange	*	*	*	*	*	*
Interactive Grouping	Y	N	N	Y	Y	Y
Scorecard	Y	N	N	Y	Y	Y
Reject Inference	Y	N	N	Y	Y	Y
*** The node does not produce this type of score code.**						

Chapter 4

Publishing the Scoring Model Files

Overview of the Publishing Process

The %INDTD_PUBLISH_MODEL macro creates the files that are needed to build the scoring functions and publishes these files to a specified database in the Teradata EDW. Only the EM_ output variables are published as Teradata scoring functions. For more information about the EM_ output variables, see “Fixed Variable Names” on page 13.

The %INDTD_PUBLISH_MODEL macro uses some of the files that are created by the SAS Enterprise Miner Score Code Export node: the scoring model program (score.sas file), the properties file (score.xml file), and, if the training data includes SAS user-defined formats, a format catalog.

The %INDTD_PUBLISH_MODEL macro takes the score.sas and score.xml files and produces the set of .c and .h files. These .c and .h files are necessary to build separate scoring functions for each of a fixed set of quantities that can be computed by the scoring model code.

If a format catalog is available, the %INDTD_PUBLISH_MODEL macro processes the format catalog and creates an .h file with C structures, which are also necessary to build the scoring functions.

This macro also produces a script of the Teradata commands that are used to register the scoring functions on the Teradata EDW.

After this script is created, the macro uses SAS/ACCESS Interface to Teradata to run the script and publish the scoring model files to the Teradata EDW.

Note: If your model generates a .c file that exceeds 4MB, the %INDTD_PUBLISH_MODEL macro generates a warning that the scoring functions might not build correctly. If you are using Teradata version V2R6 or TD12 on Linux, you can increase the size of the scoring function by installing a Teradata patch. For more information, see the *SAS In-Database Products: Administrator's Guide*.

Running the %INDTD_PUBLISH_MODEL Macro

Macro Run Process

To run the %INDTD_PUBLISH_MODEL macro, complete the following steps:

1. Create a scoring model using SAS Enterprise Miner.
2. Use the SAS Enterprise Miner Score Code Export node to create a score output directory and populate the directory with the score.sas file, the score.xml file, and, if needed, the format catalog.
3. Test your connection to Teradata with a local utility such as BTEQ.
4. Start SAS 9.2 and submit the following commands in the Program Editor or Enhanced Editor:

   ```
   %indtdpm;
   %let indconn = server="myserver" user="myuserid" password="xxxx"
       database="mydb";
   ```

   The %INDTDPM macro searches the autocall library for the indtdpm.sas file. The indtdpm.sas file contains all the macro definitions that are used in conjunction with the %INDTD_PUBLISH_MODEL macro. The indtdpm.sas file should be in one of the directories listed in the SASAUTOS= system option in your configuration file. If the indtdpm.sas file is not present, the %INDTDPM macro call (%INDTDPM; statement) issues the following message:

   ```
   macro indtdpm not defined
   ```

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

   Here is the syntax for the value of the INDCONN macro variable:

   SERVER="*server* " USER="*user* " PASSWORD="*password* " DATABASE="*database*";

   TIP The INDCONN macro variable is not passed as an argument to the %INDTD_PUBLISH_MODEL macro. This information can be concealed in your SAS job. For example, you can place it in an autoexec file and apply permissions on that file so that others cannot access the user credentials.

5. Run the %INDTD_PUBLISH_MODEL macro.

   Messages are written to the SAS log that indicate the success or failure of the creation of the scoring functions.

   For more information, see “%INDTD_PUBLISH_MODEL Macro Syntax” on page 21.

Note: USER librefs that are not assigned to the WORK library might cause unexpected or unsuccessful behavior.

%INDTD_PUBLISH_MODEL Macro Syntax

%INDTD_PUBLISH_MODEL
(DIR=*input-directory-path*, MODELNAME=*name*
<, DATASTEP=*score-program-filename*>
<, XML=*xml-filename*>
<, DATABASE=*database-name*>
<, FMTCAT=*format-catalog-filename*>
<, ACTION=CREATE | REPLACE | DROP>
<, MODE=PROTECTED | UNPROTECTED>
<, OUTDIR=*diagnostic-output-directory*>
);

Arguments

DIR=*input-directory-path*
specifies the directory where the scoring model program, the properties file, and the format catalog are located.

This is the directory that is created by the SAS Enterprise Miner Score Code Export node. This directory contains the score.sas file, the score.xml file, and, if user-defined formats were used, the format catalog.

Requirement: You must use a fully qualified pathname.

Interaction: If you do not use the default directory that is created by SAS Enterprise Miner, you must specify the DATASTEP=, XML=, and, if needed, FMTCAT= arguments.

See: "Special Characters in Directory Names" on page 24

MODELNAME=*name*
specifies the name that is prepended to each output function to ensure that each scoring function name is unique on the Teradata database.

Restriction: The scoring function name is a combination of the model and the output variable names. The scoring function name cannot exceed 30 characters. For more information, see "Scoring Function Names" on page 27.

Requirement: The model name must be a valid SAS name that is ten characters or fewer. For more information about valid SAS names, see the topic on rules for words and names in *SAS 9.2 Language Reference: Concepts*.

Interaction: Only the EM_ output variables are published as Teradata scoring functions. For more information about the EM_ output variables, see "Fixed Variable Names" on page 13 and "Scoring Function Names" on page 27.

DATASTEP=*score-program-filename*
specifies the name of the scoring model program file that was created by using the SAS Enterprise Miner Score Code Export node.

Default: score.sas

Restriction: Only DATA step programs that are produced by the SAS Enterprise Miner Score Code Export node can be used.

Interaction: If you use the default score.sas file that is created by the SAS Enterprise Miner Score Code Export node, you do not need to specify the DATASTEP= argument.

XML=*xml-filename*
specifies the name of the properties XML file that was created by the SAS Enterprise Miner Score Code Export node.

Default: score.xml

Restriction: Only XML files that are produced by the SAS Enterprise Miner Score Code Export node can be used.

Restriction: The maximum number of output variables is 128.

Interaction: If you use the default score.xml file that is created by the SAS Enterprise Miner Score Code Export node, you do not need to specify the XML= argument.

DATABASE=*database-name*
specifies the name of a Teradata database to which the scoring functions and formats are published.

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 "Macro Run Process" on page 20.

TIP You can publish the scoring functions and formats to a shared database where other users can access them.

FMTCAT=*format-catalog-filename*
specifies the name of the format catalog file that contains all user-defined formats that were created by the FORMAT procedure and that are referenced in the DATA step scoring model program.

Restriction: Only format catalog files that are produced by the SAS Enterprise Miner Score Code Export node can be used.

Interaction: If you use the default format catalog that is created by the SAS Enterprise Miner Score Code Export node, you do not need to specify the FMTCAT= argument.

Interaction: If you do not use the default catalog name (FORMATS) or the default library (WORK or LIBRARY) when you create user-defined formats, you must use the FMTSEARCH system option to specify the location of the format catalog. For more information, see PROC FORMAT in the *Base SAS 9.2 Procedures Guide*.

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

CREATE	creates a new function.
REPLACE	overwrites the current function, if a function by the same name is already registered.
DROP	causes all functions for this model to be dropped from the Teradata database.

Default: CREATE

TIP If the function has been previously defined and you specify ACTION=CREATE, you will receive warning messages from Teradata. If the function has been previously defined and you specify ACTION=REPLACE, no warnings are issued.

MODE=PROTECTED | UNPROTECTED
specifies whether the running code is isolated in a separate process in the Teradata database so that a program fault will not cause the database to stop.

Default: PROTECTED

TIP After a function is validated in PROTECTED mode, it can be republished in UNPROTECTED mode. This could result in a significant performance gain.

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

Files that are produced include an event log that contains detailed information about the success or failure of the publishing process and sample SQL code (SampleSQL.txt). For more information about the SampleSQL.txt file, see "Scoring Function Names" on page 27.

Tip: This argument is useful when testing your scoring models.

See: "Special Characters in Directory Names" on page 24

Modes of Operation

The %INDTD_PUBLISH_MODEL macro has two modes of operation: protected and unprotected. You specify the mode by setting the MODE= argument.

The default mode of operation is protected. Protected mode means that the macro code is isolated in a separate process in the Teradata database, and any error does not cause the database to stop. It is recommended that you run the %INDTD_PUBLISH_MODEL macro in protected mode during acceptance tests.

When the %INDTD_PUBLISH_MODEL macro is ready for production, you can run the macro in unprotected mode. Note that you could see a performance advantage when you run in unprotected mode.

Model Publishing Example

```
%indtdpm;
%let indconn = server="terabase" user="user1" password="open1" database="mydb";
%indtd_publish_model( dir=C:\SASIN\baseball1, modelname=baseball1);
```

This sequence of macros generates a separate .c file for each output parameter of interest. Each output stub calls into a shared scoring main which is compiled first. The %INDTD_PUBLISH_MODEL macro also produces a text file of Teradata CREATE FUNCTION commands as shown in the following example.

Note: This file is shown for illustrative purposes. The text file that is created by the %INDTD_PUBLISH_MODEL macro cannot be viewed and is deleted after the macro is complete.

```
CREATE FUNCTION baseball1_EM_eventprobablility
(
"CR_ATBAT" float,
"CR_BB" float,
"CR_HITS" float,
"CR_HOME" float,
"CR_RBI" float,
"CR_RUNS" float,
"DIVISION" varchar(31),
"LEAGUE" varchar(31),
"NO_ASSTS" float,
"NO_ATBAT" float,
"NO_BB" float,
"NO_ERROR" float,
"NO_HITS" float,
"NO_HOME" float,
"NO_OUTS" float,
"NO_RBI" float,
"NO_RUNS" float,
"YR_MAJOR" float
)
RETURNS float
LANGUAGE C
NO SQL
PARAMETER STYLE SQL
NOT DETERMINISTIC
CALLED ON NULL INPUT
EXTERNAL NAME 'SL!"jazxfbrs"'
'!CI!tkcsparm!c:\SASIN\baseball1\tkcsparm.h'
'!CS!baseball1_EM_eventprobability!c:\SASIN\baseball1\EM_eventprobability.c';
```

After the scoring functions are installed, they can be invoked in Teradata using SQL, as illustrated in the following example. Each output value is created as a separate function call in the select list.

```
select baseball1_EM_eventprobability
(
"CR_ATBAT",
"CR_BB",
"CR_HITS",
"CR_HOME",
"CR_RBI",
"CR_RUNS",
"DIVISION",
"LEAGUE",
"NO_ASSTS",
"NO_ATBAT",
"NO_BB",
"NO_ERROR",
"NO_HITS",
"NO_HOME",
"NO_OUTS"
) as homeRunProb from MLBTera;
```

Special Characters in Directory Names

If the directory names that are used in the macros contain any of the following special characters, you must mask the characters by using the %STR macro quoting function. For more information, see the %STR function and macro string quoting topic in *SAS Macro Language: Reference*.

Character	How to Represent
blank[1]	%str( )
*[2]	%str(*)

Character	How to Represent
;	%str(;)
,	%str(,)
=	%str(=)
+	%str(+)
-	%str(-)
>	%str(>)
<	%str(<)
^	%str(^)
|	%str(|)
&	%str(&)
#	%str(#)
/	%str(/)
~	%str(~)
%	%str(%%)
'	%str(%')
"	%str(%")
(	%str(%()
)	%str(%))
¬	%str(¬)

[1]Only leading blanks require the %STR function, but you should avoid using leading blanks in directory names.

[2]Asterisks (*) are allowed in UNIX directory names. Asterisks are not allowed in Windows directory names. In general, you should avoid using asterisks in directory names.

Here are some examples of directory names with special characters:

Directory	Code representation
`c:\temp\Sales(part1)`	`c:\temp\Sales%str(%()part1%str(%))`
`c:\temp\Drug “trial” X`	`c:\temp\Drug %str(%")trial(%str(%") X`

Directory	Code representation
`c:\temp\Disc's 50% Y`	`c:\temp\Disc%str(%')s 50%str(%%) Y`
`c:\temp\Pay,Emp=Z`	`c:\temp\Pay%str(,)Emp%str(=)Z`

Teradata Permissions

Because functions are associated with a database, the functions inherit the access rights of that database. It could be useful to create a separate shared database for scoring functions so that access rights can be customized as needed.

In addition, to publish the scoring functions in Teradata, you must have the following permissions on the database where the functions are published:

CREATE FUNCTION
DROP FUNCTION
EXECUTE FUNCTION
ALTER FUNCTION

To obtain database permissions, contact your database administrator.

Chapter 5

Scoring Functions Inside the Teradata EDW

Scoring Function Names

The names of the scoring functions that are built in Teradata have the following format:

```
modelname_EM_outputvarname
```

modelname is the name that was specified in the MODELNAME argument of the %INDTD_PUBLISH_MODEL macro. *modelname* is always followed by _EM_ in the scoring function name. For more information about the MODELNAME argument, see "Running the %INDTD_PUBLISH_MODEL Macro" on page 20.

outputvarname is derived from the names of the EM_ output variables in the score.xml file that is generated from the SAS Enterprise Miner Score Code Export node. For more information about the score.xml file, see "Fixed Variable Names" on page 13.

One scoring function is created for each EM_ output variable in the score.xml file. For example, if the scoring model DATA step program takes ten inputs and creates three new variables, then three scoring functions are defined, each with the name of an output variable. For example, if you set MODELNAME=credit in the %INDTD_PUBLISH_MODEL macro, and the EM_ output variables are "EM_PREDICTION", "EM_PROBABILITY", and "EM_DECISION", then the name of the scoring functions that are created would be "credit_EM_PREDICTION", "credit_EM_PROBABILITY", and "credit_EM_DECISION".

Note: The scoring function name cannot exceed 30 characters.

CAUTION:

When the scoring function is generated, the names are case-insensitive. Consequently, if you have model names "Model01" and "model01", and you create two scoring functions, the second scoring function will overwrite the first scoring function.

Using the Scoring Functions

The scoring functions are available to use in any SQL expression in the same way that Teradata built-in functions are used. For an example, see "Model Publishing Example" on page 23.

There are four ways to see the scoring functions that are created:

- From Teradata, you can log on to the database using a client tool such as BTEQ and submit an SQL statement. The following example assumes that the model name that you used to create the scoring functions is **mymodel**.

  ```
  bteq .logon myserver/myuserid,mypassword;
    select * from dbc.tables where tablename like '%mymodel%';
  ```

- From SAS you can use SQL procedure code that produces output in the LST file. The following example assumes that the model name that you used to create the scoring functions is **mymodel**.

  ```
  proc sql noerrorstop;
    connect to teradata (user=user password=pass server=server);
    select *
       from connection to teradata
       (select tablename,tablekind,databasename,LastALterTimeStamp
        from dbc.tables where
        databasename='sas' and tablename like '%mymodel%'
        and tablekind='F');

  disconnect teradata;
  quit;
  ```

  You can also use the SASTRACE and SASTRACELOC system options to generate tracing information. For more information about these system options, see the *SAS 9.2 Language Reference: Dictionary*.

- You can look at the SampleSQL.txt file that is produced when the %INDTD_PUBLISH_MODEL macro is successfully run. This file can be found in the output directory (OUTDIR argument) that you specify in the macro.

  The SampleSQL.txt file contains basic code that, with modifications, can be used to run your score code inside Teradata.

  For example, the SampleSQL.txt file refers to an ID column in **allmush1_intab** that is populated with a unique integer from 1 to *n*, with *n* being the number of rows in the table. The ID column uniquely identifies each row. You would replace the ID column with your own primary key column.

  Note: The function and table names must be fully qualified if the functions and tables are not in the same database.

  The following example assumes that the model name that you used to create the scoring functions is **allmush1**.

  ```
     drop table allmush1_outtab;
     create table allmush1_outtab(
      id integer
     ,"EM_CLASSIFICATION" varchar(33)
     ,"EM_EVENTPROBABILITY" float
  ```

```
,"EM_PROBABILITY" float
);
insert into allmush1_outtab(
 id
,"EM_CLASSIFICATION"
,"EM_EVENTPROBABILITY"
,"EM_PROBABILITY"
)
select id,
 allmush1_em_classification("BRUISES"
,"CAPCOLOR"
,"GILLCOLO"
,"GILLSIZE"
,"HABITAT"
,"ODOR"
,"POPULAT"
,"RINGNUMB"
,"RINGTYPE"
,"SPOREPC"
,"STALKCBR"
,"STALKROO"
,"STALKSAR"
,"STALKSHA"
,"VEILCOLO")
  as "EM_CLASSIFICATION",
 allmush1_em_eventprobability("BRUISES"
,"CAPCOLOR"
,"GILLCOLO"
,"GILLSIZE"
,"HABITAT"
,"ODOR"
,"POPULAT"
,"RINGNUMB"
,"RINGTYPE"
,"SPOREPC"
,"STALKCBR"
,"STALKROO"
,"STALKSAR"
,"STALKSHA"
,"VEILCOLO")
  as "EM_EVENTPROBABILITY",
 allmush1_em_probability("BRUISES"
,"CAPCOLOR"
,"GILLCOLO"
,"GILLSIZE"
,"HABITAT"
,"ODOR"
,"POPULAT"
,"RINGNUMB"
,"RINGTYPE"
,"SPOREPC"
,"STALKCBR"
,"STALKROO"
,"STALKSAR"
,"STALKSHA"
,"VEILCOLO")
```

```
   as "EM_PROBABILITY"
 from allmush1_intab ;
```

- You can look at the SAS log. A message is printed to the SAS log that states whether a scoring function is successfully or not successfully created or replaced.

Chapter 6
SAS Scoring Accelerator and SAS Model Manager

Using the SAS Scoring Accelerator with SAS Model Manager

You can use SAS Scoring Accelerator in conjunction with the SAS Model Manager to manage and deploy scoring models in Teradata.

The **Publish Scoring Function** of SAS Model Manager enables you to publish classification and prediction model types to the Teradata EDW. When you publish a Teradata scoring function for a project, SAS Model Manager exports the project's champion model to the SAS Metadata Repository and calls the SAS Scoring Accelerator to create the scoring functions. The scoring functions are deployed inside Teradata based on the project's champion model score code. The scoring function is then validated automatically against a default train table to ensure that the scoring results are correct. A scoring application (for example, a call center application that calls the SAS Model Manager Java Scoring API) can then execute the scoring functions in the Teradata EDW. The scoring functions extend the Teradata SQL language and can be used on SQL statements like other Teradata functions.

For more information, see the *SAS Model Manager: User's Guide*.

Index