THE POWER TO KNOW.

SAS® Scoring Accelerator 3.1 for Aster *n*Cluster

User's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc 2011. *SAS® Scoring Accelerator 3.1 for Aster nCluster*. Cary, NC: SAS Institute Inc.

SAS® Scoring Accelerator 3.1 for Aster *n*Cluster

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

1st electronic book, February 2011

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

Chapter 1 • Introduction to the SAS Scoring Accelerator for Aster nCluster **1**
Overview of the SAS Scoring Accelerator for Aster nCluster 1
How the SAS Scoring Accelerator for Aster nCluster Works 1

Chapter 2 • Deployed Components for In-Database Processing **3**
Overview of Deployed Components for In-Database Processing 3

Chapter 3 • Exporting the Scoring Model Files from SAS Enterprise Miner **5**
Overview of the Score Code Export Node 5
Using the Score Code Export Node Compared with Registering Models on the SAS Metadata Server 6
Using the Score Code Export Node 6
Output Created by the Score Code Export Node 8

Chapter 4 • Publishing the Scoring Files **17**
Overview of the Publishing Process 17
Running the %INDAC_PUBLISH_MODEL Macro 18
Special Characters in Directory Names 21

Chapter 5 • Scoring Files and Functions Inside the Aster nCluster Database **25**
Aster nCluster Scoring Files 25
SAS_SCORE() Function 27

Appendix 1 • Scoring File Examples **29**
Example of a .ds2 Scoring File 29
Example of an Input and Output Variables Scoring File 49
Example of a User-Defined Formats Scoring File 56

Index **63**

Chapter 1

Introduction to the SAS Scoring Accelerator for Aster *n*Cluster

Overview of the SAS Scoring Accelerator for Aster nCluster 1

How the SAS Scoring Accelerator for Aster nCluster Works 1

Overview of the SAS Scoring Accelerator for Aster *n*Cluster

When using conventional processing to access data inside an Aster *n*Cluster database, 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 Aster *n*Cluster database. 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 Aster *n*Cluster database to the SAS scoring process increases.

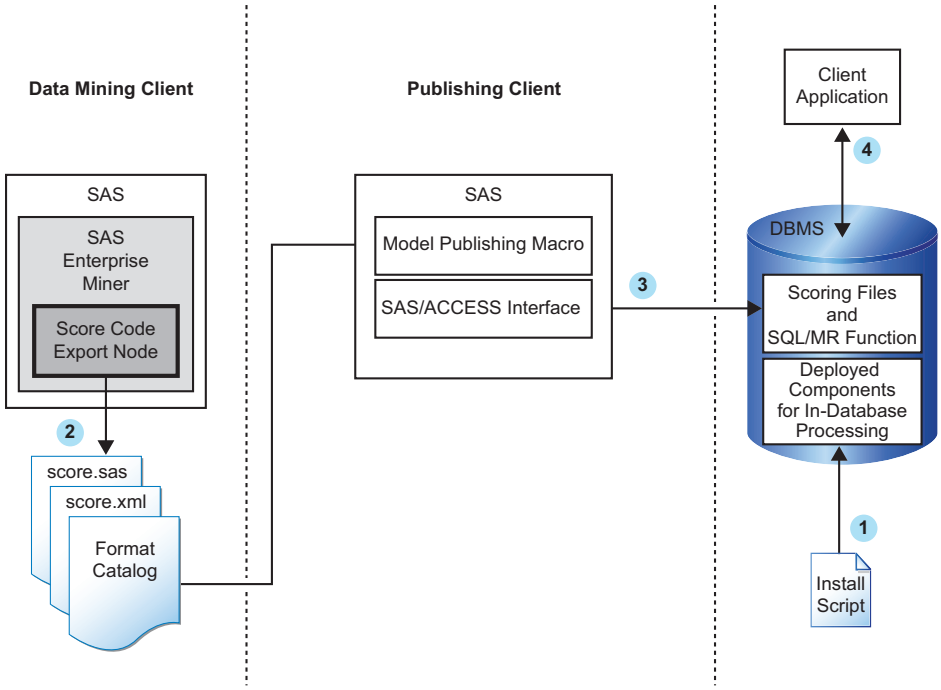
The SAS Scoring Accelerator for Aster *n*Cluster embeds the robustness of SAS Enterprise Miner scoring models directly in the highly scalable Aster *n*Cluster database. When you use the SAS In-Database technology and the SAS Scoring Accelerator for Aster *n*Cluster, the scoring processing is done inside the database, and thus does not require the transfer of data.

The SAS Scoring Accelerator for Aster *n*Cluster takes the models that are developed by SAS Enterprise Miner and translates them into scoring files that can be deployed inside Aster *n*Cluster. After the scoring files are published, a SAS SQL/Map Reduce (SQL/MR) function executes the scoring model.

How the SAS Scoring Accelerator for Aster *n*Cluster Works

Using SAS Enterprise Miner, you can generate SAS DATA step code that contains scoring functions. The SAS Scoring Accelerator for Aster *n*Cluster takes the SAS DATA step 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 Aster *n*Cluster database as scoring files. Inside the Aster *n*Cluster database, a SAS SQL/MR function is installed and used to execute the scoring model. 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 Aster *n*Cluster database.

   For more information, see Chapter 2, "Deployed Components for In-Database Processing," on page 3.

   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 files to a score output directory.

   For more information, see Chapter 3, "Exporting the Scoring Model Files from SAS Enterprise Miner," on page 5.

3 Start SAS 9.2 and run the SAS publishing macros, which create the scoring files that are needed to execute the scoring model and publish those files to the Aster *n*Cluster database.

   For more information, see Chapter 4, "Publishing the Scoring Files," on page 17.

4 After the scoring files are created, they are available for use by a SAS SQL/MR function in the same way that the Aster *n*Cluster SQL/MR functions are used.

   For more information, see Chapter 5, "Scoring Files and Functions Inside the Aster nCluster Database," on page 25.

Chapter 2

Deployed Components for In-Database Processing

Overview of Deployed Components for In-Database Processing 3

Overview of Deployed Components for In-Database Processing

Components that are deployed to Aster *n*Cluster for in-database processing are contained in a self-extracting TAR file (accelastrfmt.sh).

The following components are deployed:

- the SAS 9.2 Formats Library for Aster *n*Cluster. The library processes any user-defined formats that might be included in your scoring model.
- the SAS Embedded Process. The SAS Embedded Process enables the scoring model to be executed directly in Aster *n*Cluster using a SAS SQL/MR function.
- the binary file for the SAS_SCORE() SQL/MR function. The SAS_SCORE() function executes the scoring model in the SAS Embedded Process.
- other run-time libraries and macros that enable you to publish your scoring model.

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 **5**

Using the Score Code Export Node Compared with Registering Models on the SAS Metadata Server **6**

Using the Score Code Export Node **6**
Using the Score Code Export Node in a Process Flow Diagram 6
Score Code Export Node Properties 7

Output Created by the Score Code Export Node **8**
Results Window 8
Output Files 9
Output Variables 10
Fixed Variable Names 11
SAS Enterprise Miner Tools Production of Score Code 12

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. Then the model can be 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 score code (score.xml).
- if the scoring program contains user-defined formats, a format catalog.
- an XML file containing descriptions of the final variables that are created by the score 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 used to develop the score code. 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 9. 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 Aster *n*Cluster. 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 12.

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

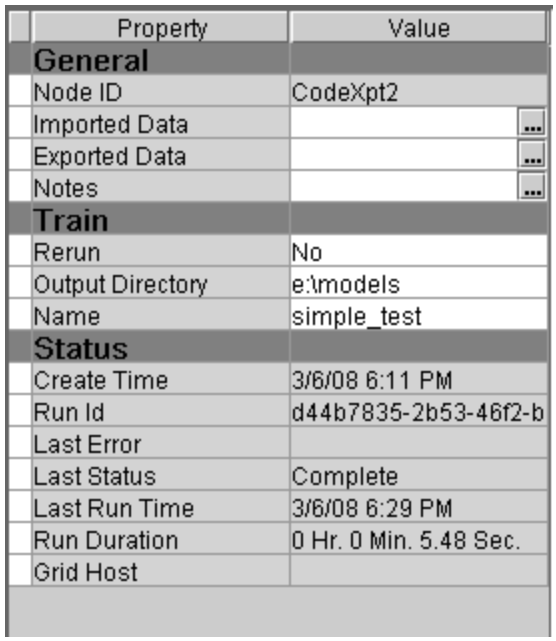
1. Select the Score node. Ensure that the following properties are set to their default value of Yes:
   - **Use Output Fixed Names**
   - **C Score**
2. Select the Score Code Export node and set the properties. The **Output Directory** property specifies the directory to store the export files. The **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 7.

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 appears and indicates 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 8.

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*.

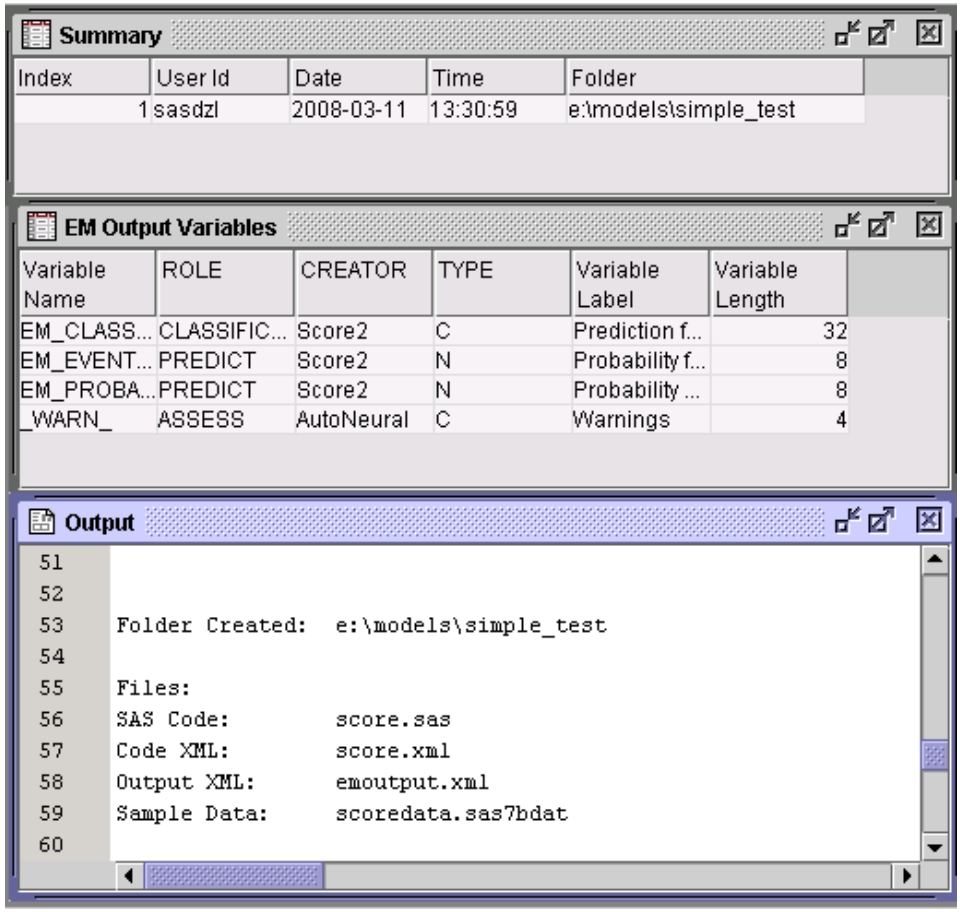
The General and Status properties for the Score Code Export node function just as they do for other nodes.

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*



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 %INDAC_PUBLISH_MODEL macro that creates the scoring files that are needed to execute the scoring model.

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 11. *Note:* The emoutput.xml file is not used by the %INDAC_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 %INDAC_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 %INDAC_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	Aster *n*Cluster
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	Aster *n*Cluster
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	Aster *n*Cluster
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	Aster *n*Cluster
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 Files

Overview of the Publishing Process **17**

Running the %INDAC_PUBLISH_MODEL Macro **18**
%INDAC_PUBLISH_MODEL Macro Run Process 18
%INDAC_PUBLISH_MODEL Macro Syntax 19
Model Publishing Macro Example 21

Special Characters in Directory Names **21**

Overview of the Publishing Process

The integration of the SAS Embedded Process and Aster *n*Cluster allows scoring code to be executed directly using the SAS Embedded Process in Aster *n*Cluster through a SQL/MR function.

The SQL/MR function is the framework for enabling execution of user-defined functions within Aster *n*Cluster through an SQL interface. A new SAS SQL/MR function, SAS_SCORE(), supports model publishing in Aster *n*Cluster.

The %INDAC_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 %INDAC_PUBLISH_MODEL macro performs the following tasks:

- takes the score.sas and score.xml files that are created using the Score Code Export node and produces two files for each scoring model. The following files are produced:
  - sasscore_*modelname*.ds2, which contains code that is executed by the SAS_SCORE() function
  - sasscore_*modelname*_io.xml, which contains the scoring model's input and output variables
- takes the format catalog, if available, and produces the sasscore_*modelname*_ufmt.xml file, which contains user-defined formats for the scoring model that is being published.
- uses the SAS/ACCESS Interface to Aster *n*Cluster to insert the three scoring files into the NC_INSTALLED_FILES table under the PUBLIC schema.

After the scoring files are published, you can call the SAS_SCORE() function to execute the scoring model. For more information, see "SAS_SCORE() Function" on page 27.

Running the %INDAC_PUBLISH_MODEL Macro

%INDAC_PUBLISH_MODEL Macro Run Process

To run the %INDAC_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. Start SAS 9.2 and submit the following commands in the Program Editor or Enhanced Editor:

   ```
   %indacpm;
   %let indconn = user=myuserid password=XXXX
   dsn=ncluster | server=myserver database=mydatabase;
   ```

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

   ```
   macro indacpm not defined
   ```

   The INDCONN macro variable is used to provide credentials to connect to Aster *n*Cluster. The value of the INDCONN macro variable for the %INDAC_PUBLISH_MODEL macro has this format:

   USER=*user* PASSWORD=*password connection-string*

   The *connection–string* argument can be expressed in either of the following forms:

   - DSN=*dsnname*
   - SERVER=*servername* DATABASE=*databasename*

   You must specify user, password, and either a DSN name or a server and database name.

   Note: You can use only PASSWORD= or PW= for the password argument. Other aliases such as PASS= or PWD= are not supported and cause errors.

   You must assign the INDCONN macro variable before the %INDAC_PUBLISH_MODEL macro is invoked.

   TIP The INDCONN macro variable is not passed as an argument to the %INDAC_PUBLISH_MODEL macro. This information can be concealed in your SAS job. You might want to place it in an autoexec file and set the permissions on the file so that others cannot access the user ID and password.

4. Run the %INDAC_PUBLISH_MODEL macro.

   Messages are written to the SAS log that indicate the success or failure of the creation of the .ds2 and .xml scoring files.

For more information, see "%INDAC_PUBLISH_MODEL Macro Syntax" on page 19.

%INDAC_PUBLISH_MODEL Macro Syntax

%INDAC_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>
<, 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 filenames that are created by SAS Enterprise Miner, you must specify the DATASTEP=, XML=, and, if needed, FMTCAT= arguments.

See: "Special Characters in Directory Names" on page 21.

MODELNAME=*name*
specifies the name that becomes part of the .ds2 and .xml scoring filenames.

Restriction: The names of the .ds2 and .xml scoring files are a combination of the model and type of filenames. A scoring filename cannot exceed 63 characters. For more information, see "Aster nCluster Scoring Files" on page 25.

Requirement: The name must be a valid SAS name. 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 in the sasscore_*modelname*_io.xml file. For more information about the EM_ output variables, see "Fixed Variable Names" on page 11 and "Aster nCluster Scoring Files" on page 25.

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.

Interaction: The SAS file that is specified in the DATASTEP= argument is translated by the %INDAC_PUBLISH_MODEL macro into the sasscore_*modelname*.ds2 file and stored in the NC_INSTALLED_FILES table under the PUBLIC schema.

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.

Interaction: The XML file is renamed to sasscore_*modelname*_io.xml by the %INDAC_PUBLISH_MODEL macro and the file is stored in the NC_INSTALLED_FILES table under the PUBLIC schema.

DATABASE=*database-name*
specifies the name of an Aster *n*Cluster 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 "%INDAC_PUBLISH_MODEL Macro Run Process" on page 18.

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

FMTCAT=*format-catalog-filename*
specifies the name of the format catalog file. The file 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 the sasscore_*modelname*.ds2, sasscore_*modelname*_io.xml, and sasscore_*modelname*_ufmt.xml files.

REPLACE overwrites the current sasscore_*modelname*.ds2, sasscore_*modelname*_io.xml, and sasscore_*modelname*_ufmt.xml files, if those files by the same name are already registered.

DROP causes the sasscore_*modelname*.ds2, sasscore_*modelname*_io.xml, and sasscore_*modelname*_ufmt.xml files to be dropped from the NC_INSTALLED_FILES table in the Aster *n*Cluster database.

Default: CREATE

TIP If the scoring files have been previously defined and you specify ACTION=CREATE, you will receive warning messages from Aster *n*Cluster. If the scoring files have been previously defined and you specify ACTION=REPLACE, no warnings are issued.

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 "Aster nCluster Scoring Files" on page 25.

TIP This argument is useful to debug a scoring model that fails to be published.

See: "Special Characters in Directory Names" on page 21

Model Publishing Macro Example

```
%indacpm;
%let indconn = server=acbase user=user1 password=open1 database=nc3875;
%indac_publish_model( dir=C:\SASIN\score, modelname=score);
```

The %INDAC_PUBLISH_MODEL macro produces these three files:

- sasscore_score.ds2. For an example, see "Example of a .ds2 Scoring File" on page 29.
- sasscore_score_io.xml. For an example, see "Example of an Input and Output Variables Scoring File" on page 49.
- sasscore_score_ufmt.xml. For an example, see "Example of a User-Defined Formats Scoring File" on page 56.

After the scoring files are installed, they can be invoked in Aster *n*Cluster using the SAS_SCORE() function. For more information, see "SAS_SCORE() Function" on page 27.

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`

Chapter 5

Scoring Files and Functions Inside the Aster *n*Cluster Database

Aster nCluster Scoring Files **25**

SAS_SCORE() Function **27**
Overview of the SAS_SCORE() Function 27
Using the SAS_SCORE() Function 27

Aster *n*Cluster Scoring Files

The %INDAC_PUBLISH_MODEL macro produces three scoring files for each model:

- sasscore_*modelname*.ds2, which contains code that is executed by the SAS_SCORE() function
- sasscore_*modelname*_io.xml, which contains the scoring model's input and output variables
- sasscore_*modelname*_ufmt.xml, which contains user-defined format's for the scoring model that is being published

These files are inserted into the NC_INSTALLED_FILES table under the PUBLIC schema. See Appendix 1, “Scoring File Examples,” on page 29 for an example of each of these files.

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

- You can log on to the database using the Aster *n*Cluster command line processor and submit an SQL statement. The following example assumes that the model name that you used to create the scoring files is **reg**.

  ```
  >act -h hostname -u username -w password -d databasename
  >select name from public.nc_installed_files where name like 'sasscore_reg%';
  ```

  Three files are listed for each model:

  ```
      name
  -------------------------
   sasscore_reg.ds2
   sasscore_reg_io.xml
   sasscore_reg_ufmt.xml
  ```

- 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 **reg**.

```
proc sql noerrorstop;
  connect to aster (user=username password=password dsn=dsnname);

select *
   from connection to aster
      (select name, owner, uploadtime
         from public.nc_installed_files where
            name like 'sasscore_reg%');
   disconnect from aster;
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 %INDAC_PUBLISH_MODEL macro is successfully run. This file can be found in the output directory (OUTDIR argument) that you specify in the %INDAC_PUBLISH_MODELmacro.

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

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

  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.

  The following example assumes that the model name that you used is **reg**.

  ```
  drop table score_outtab;
  create table score_outtab(
   id integer
  ,"EM_CLASSIFICATION" varchar(256)
  ,"EM_EVENTPROBABILITY" float
  ,"EM_PROBABILITY" float
  );
  insert into score_outtab(
   id
  ,"EM_CLASSIFICATION"
  ,"EM_EVENTPROBABILITY"
  ,"EM_PROBABILITY"
  )
  select id,
  "EM_CLASSIFICATION",
  "EM_EVENTPROBABILITY",
  "EM_PROBABILITY"
  from sas_score(on score_intab model('reg'));
  ```

- You can look at the SAS log. A message that indicates whether the scoring files are successfully or not successfully created is printed to the SAS log.

SAS_SCORE() Function

Overview of the SAS_SCORE() Function

The SAS_SCORE() function executes the scoring model using the SAS Embedded Process in Aster *n*Cluster. The SAS_SCORE() function is deployed and stored in the NC_INSTALLED_FILES table under the PUBLIC schema during the installation and configuration of the SAS Scoring Accelerator for Aster *n*Cluster. For more information, see "Overview of Deployed Components for In-Database Processing" on page 3.

Using the SAS_SCORE() Function

The SAS_SCORE() function is available to use in the FROM clause in any SQL expression in the same way that Aster *n*Cluster SQL/MR functions are used.

The syntax of the SAS_SCORE() function is as follows:

FROM **SAS_SCORE**(ON *input-table* MODEL('*model-name*'))

Arguments

input-table
specifies the input table that is used by the SAS_SCORE() function.

model-name
specifies the name of the model. The value of this argument is the same as the value of MODELNAME=*name* argument for the %INDAC_PUBLISH_MODEL macro.

Here is an example of using the SAS_SCORE function. In this example, the input table is **score_intab** and the model name is **reg**.

```
select id, em_classification, em_eventprobability, em_probability
   from sas_score (on score_intab model('reg'));
```

Appendix 1

Scoring File Examples

Example of a .ds2 Scoring File 29
Example of an Input and Output Variables Scoring File 49
Example of a User-Defined Formats Scoring File 56

Example of a .ds2 Scoring File

This is an example of a .ds2 scoring file. The filename is sasscore_score.ds2.

```
data &ASTER_OUTPUT;
 #_local _LPMAX;
 #_local _P4;
 #_local _P3;
 #_local _P2;
 #_local _P1;
 #_local _P0;
 #_local _IY;
 #_local _MAXP;
 #_local _LP3;
 #_local _LP2;
 #_local _LP1;
 #_local _LP0;
 #_local _TEMP;
 #_local _7_1;
 #_local _7_0;
 #_local _6_2;
 #_local _6_1;
 #_local _6_0;
 #_local _5_14;
 #_local _5_13;
 #_local _5_12;
 #_local _5_11;
 #_local _5_10;
 #_local _5_9;
 #_local _5_8;
 #_local _5_7;
 #_local _5_6;
 #_local _5_5;
 #_local _5_4;
```

```
 #_local _5_3;
 #_local _5_2;
 #_local _5_1;
 #_local _5_0;
 #_local _3_10;
 #_local _3_9;
 #_local _3_8;
 #_local _3_7;
 #_local _3_6;
 #_local _3_5;
 #_local _3_4;
 #_local _3_3;
 #_local _3_2;
 #_local _3_1;
 #_local _3_0;
 #_local _2_12;
 #_local _2_11;
 #_local _2_10;
 #_local _2_9;
 #_local _2_8;
 #_local _2_7;
 #_local _2_6;
 #_local _2_5;
 #_local _2_4;
 #_local _2_3;
 #_local _2_2;
 #_local _2_1;
 #_local _2_0;
 #_local _DM_FIND;
 #_local _1_14;
 #_local _1_13;
 #_local _1_12;
 #_local _1_11;
 #_local _1_10;
 #_local _1_9;
 #_local _1_8;
 #_local _1_7;
 #_local _1_6;
 #_local _1_5;
 #_local _1_4;
 #_local _1_3;
 #_local _1_2;
 #_local _1_1;
 #_local _1_0;
 #_local _DM_BAD;
dcl char(4) _WARN_;
dcl char(6) I_ATTACK;
dcl char(6) U_ATTACK;
dcl char(32) EM_CLASSIFICATION;
dcl double COUNT;
dcl double DIF_SRVR;
dcl char(32) FLAG;
dcl double HOT;
dcl double SAM_SRAT;
dcl char(32) SERVICE;
dcl double SRV_CNT;
```

```
method run();
 dcl char(8) _NORM8;
 dcl char(8) _NORM8;
 dcl char(12) _DM12;
 dcl char(12) _DM12;
 dcl char(12) _DM12;
 dcl char(12) _DM12;
 dcl char(12) _DM12;
 dcl char(12) _DM12;
 dcl char(6) REGDRU[5];
 dcl char(6) REGDRF[5];
REGDRU:=('u2r   ', 'r2l   ', 'probe ', 'normal', 'dos   ');
REGDRF:=('U2R', 'R2L', 'PROBE', 'NORMAL', 'DOS');
set &ASTER_INPUT;
_WARN_ = ' ';
if (COUNT = .) then AOV16_COUNT = 8.0;
 else if (COUNT <= 31.9375) then AOV16_COUNT = 1.0;
 else if (COUNT <= 63.875) then AOV16_COUNT = 2.0;
 else if (COUNT <= 95.8125) then AOV16_COUNT = 3.0;
 else if (COUNT <= 127.75) then AOV16_COUNT = 4.0;
 else if (COUNT <= 159.6875) then AOV16_COUNT = 5.0;
 else if (COUNT <= 191.625) then AOV16_COUNT = 6.0;
 else if (COUNT <= 223.5625) then AOV16_COUNT = 7.0;
 else if (COUNT <= 255.5) then AOV16_COUNT = 8.0;
 else if (COUNT <= 287.4375) then AOV16_COUNT = 9.0;
 else if (COUNT <= 319.375) then AOV16_COUNT = 10.0;
 else if (COUNT <= 351.3125) then AOV16_COUNT = 11.0;
 else if (COUNT <= 383.25) then AOV16_COUNT = 12.0;
 else if (COUNT <= 415.1875) then AOV16_COUNT = 13.0;
 else if (COUNT <= 447.125) then AOV16_COUNT = 14.0;
 else if (COUNT <= 479.0625) then AOV16_COUNT = 15.0;
 else AOV16_COUNT = 16.0;
if (SRV_CNT = .) then AOV16_SRV_CNT = 7.0;
 else if (SRV_CNT <= 31.9375) then AOV16_SRV_CNT = 1.0;
 else if (SRV_CNT <= 63.875) then AOV16_SRV_CNT = 2.0;
 else if (SRV_CNT <= 95.8125) then AOV16_SRV_CNT = 3.0;
 else if (SRV_CNT <= 127.75) then AOV16_SRV_CNT = 4.0;
 else if (SRV_CNT <= 159.6875) then AOV16_SRV_CNT = 5.0;
 else if (SRV_CNT <= 191.625) then AOV16_SRV_CNT = 6.0;
 else if (SRV_CNT <= 223.5625) then AOV16_SRV_CNT = 7.0;
 else if (SRV_CNT <= 255.5) then AOV16_SRV_CNT = 8.0;
 else if (SRV_CNT <= 287.4375) then AOV16_SRV_CNT = 9.0;
 else if (SRV_CNT <= 319.375) then AOV16_SRV_CNT = 10.0;
 else if (SRV_CNT <= 351.3125) then AOV16_SRV_CNT = 11.0;
 else if (SRV_CNT <= 383.25) then AOV16_SRV_CNT = 12.0;
 else if (SRV_CNT <= 415.1875) then AOV16_SRV_CNT = 13.0;
 else if (SRV_CNT <= 447.125) then AOV16_SRV_CNT = 14.0;
 else if (SRV_CNT <= 479.0625) then AOV16_SRV_CNT = 15.0;
 else AOV16_SRV_CNT = 16.0;
if (SAM_SRAT = .) then AOV16_SAM_SRAT = 14.0;
 else if (SAM_SRAT <= 0.0625) then AOV16_SAM_SRAT = 1.0;
 else if (SAM_SRAT <= 0.125) then AOV16_SAM_SRAT = 2.0;
 else if (SAM_SRAT <= 0.1875) then AOV16_SAM_SRAT = 3.0;
 else if (SAM_SRAT <= 0.25) then AOV16_SAM_SRAT = 4.0;
 else if (SAM_SRAT <= 0.3125) then AOV16_SAM_SRAT = 5.0;
 else if (SAM_SRAT <= 0.375) then AOV16_SAM_SRAT = 6.0;
```

```
 else if (SAM_SRAT <= 0.4375) then AOV16_SAM_SRAT = 7.0;
 else if (SAM_SRAT <= 0.5) then AOV16_SAM_SRAT = 8.0;
 else if (SAM_SRAT <= 0.5625) then AOV16_SAM_SRAT = 9.0;
 else if (SAM_SRAT <= 0.625) then AOV16_SAM_SRAT = 10.0;
 else if (SAM_SRAT <= 0.6875) then AOV16_SAM_SRAT = 11.0;
 else if (SAM_SRAT <= 0.75) then AOV16_SAM_SRAT = 12.0;
 else if (SAM_SRAT <= 0.8125) then AOV16_SAM_SRAT = 13.0;
 else if (SAM_SRAT <= 0.875) then AOV16_SAM_SRAT = 14.0;
 else if (SAM_SRAT <= 0.9375) then AOV16_SAM_SRAT = 15.0;
 else AOV16_SAM_SRAT = 16.0;
if (DIF_SRVR = .) then AOV16_DIF_SRVR = 1.0;
 else if (DIF_SRVR <= 0.0625) then AOV16_DIF_SRVR = 1.0;
 else if (DIF_SRVR <= 0.125) then AOV16_DIF_SRVR = 2.0;
 else if (DIF_SRVR <= 0.1875) then AOV16_DIF_SRVR = 3.0;
 else if (DIF_SRVR <= 0.25) then AOV16_DIF_SRVR = 4.0;
 else if (DIF_SRVR <= 0.3125) then AOV16_DIF_SRVR = 5.0;
 else if (DIF_SRVR <= 0.375) then AOV16_DIF_SRVR = 6.0;
 else if (DIF_SRVR <= 0.4375) then AOV16_DIF_SRVR = 7.0;
 else if (DIF_SRVR <= 0.5) then AOV16_DIF_SRVR = 8.0;
 else if (DIF_SRVR <= 0.5625) then AOV16_DIF_SRVR = 9.0;
 else if (DIF_SRVR <= 0.625) then AOV16_DIF_SRVR = 10.0;
 else if (DIF_SRVR <= 0.6875) then AOV16_DIF_SRVR = 11.0;
 else if (DIF_SRVR <= 0.75) then AOV16_DIF_SRVR = 12.0;
 else if (DIF_SRVR <= 0.8125) then AOV16_DIF_SRVR = 13.0;
 else if (DIF_SRVR <= 0.875) then AOV16_DIF_SRVR = 14.0;
 else if (DIF_SRVR <= 0.9375) then AOV16_DIF_SRVR = 15.0;
 else AOV16_DIF_SRVR = 16.0;
if (HOT = .) then AOV16_HOT = 1.0;
 else if (HOT <= 1.875) then AOV16_HOT = 1.0;
 else if (HOT <= 3.75) then AOV16_HOT = 2.0;
 else if (HOT <= 5.625) then AOV16_HOT = 3.0;
 else if (HOT <= 7.5) then AOV16_HOT = 4.0;
 else if (HOT <= 9.375) then AOV16_HOT = 5.0;
 else if (HOT <= 11.25) then AOV16_HOT = 6.0;
 else if (HOT <= 13.125) then AOV16_HOT = 7.0;
 else if (HOT <= 15.0) then AOV16_HOT = 8.0;
 else if (HOT <= 16.875) then AOV16_HOT = 9.0;
 else if (HOT <= 18.75) then AOV16_HOT = 10.0;
 else if (HOT <= 20.625) then AOV16_HOT = 11.0;
 else if (HOT <= 22.5) then AOV16_HOT = 12.0;
 else if (HOT <= 24.375) then AOV16_HOT = 13.0;
 else if (HOT <= 26.25) then AOV16_HOT = 14.0;
 else if (HOT <= 28.125) then AOV16_HOT = 15.0;
 else AOV16_HOT = 16.0;
_NORM8 = DMNORM(SERVICE, 32.0);
select (_NORM8);
when ('IRC     ') G_SERVICE = 2.0;
when ('X11     ') G_SERVICE = 2.0;
when ('Z39_50  ') G_SERVICE = 1.0;
when ('AUTH    ') G_SERVICE = 2.0;
when ('BGP     ') G_SERVICE = 0.0;
when ('COURIER ') G_SERVICE = 1.0;
when ('CSNET_NS') G_SERVICE = 1.0;
when ('CTF     ') G_SERVICE = 0.0;
when ('DAYTIME ') G_SERVICE = 1.0;
when ('DISCARD ') G_SERVICE = 0.0;
```

```
when ('DOMAIN  ') G_SERVICE = 1.0;
when ('DOMAIN_U') G_SERVICE = 2.0;
when ('ECHO    ') G_SERVICE = 0.0;
when ('ECO_I   ') G_SERVICE = 2.0;
when ('ECR_I   ') G_SERVICE = 0.0;
when ('EFS     ') G_SERVICE = 1.0;
when ('EXEC    ') G_SERVICE = 0.0;
when ('FINGER  ') G_SERVICE = 2.0;
when ('FTP     ') G_SERVICE = 2.0;
when ('FTP_DATA') G_SERVICE = 2.0;
when ('GOPHER  ') G_SERVICE = 1.0;
when ('HOSTNAME') G_SERVICE = 0.0;
when ('HTTP    ') G_SERVICE = 2.0;
when ('HTTP_443') G_SERVICE = 0.0;
when ('IMAP4   ') G_SERVICE = 1.0;
when ('ISO_TSAP') G_SERVICE = 0.0;
when ('KLOGIN  ') G_SERVICE = 0.0;
when ('KSHELL  ') G_SERVICE = 0.0;
when ('LDAP    ') G_SERVICE = 0.0;
when ('LINK    ') G_SERVICE = 1.0;
when ('LOGIN   ') G_SERVICE = 0.0;
when ('MTP     ') G_SERVICE = 1.0;
when ('NAME    ') G_SERVICE = 0.0;
when ('NETBIOS_') G_SERVICE = 0.0;
when ('NETSTAT ') G_SERVICE = 0.0;
when ('NNSP    ') G_SERVICE = 0.0;
when ('NNTP    ') G_SERVICE = 1.0;
when ('NTP_U   ') G_SERVICE = 2.0;
when ('OTHER   ') G_SERVICE = 2.0;
when ('POP_2   ') G_SERVICE = 0.0;
when ('POP_3   ') G_SERVICE = 2.0;
when ('PRINTER ') G_SERVICE = 1.0;
when ('PRIVATE ') G_SERVICE = 1.0;
when ('RED_I   ') G_SERVICE = 2.0;
when ('REMOTE_J') G_SERVICE = 1.0;
when ('RJE     ') G_SERVICE = 1.0;
when ('SHELL   ') G_SERVICE = 0.0;
when ('SMTP    ') G_SERVICE = 2.0;
when ('SQL_NET ') G_SERVICE = 0.0;
when ('SSH     ') G_SERVICE = 1.0;
when ('SUNRPC  ') G_SERVICE = 1.0;
when ('SUPDUP  ') G_SERVICE = 1.0;
when ('SYSTAT  ') G_SERVICE = 1.0;
when ('TELNET  ') G_SERVICE = 2.0;
when ('TFTP_U  ') G_SERVICE = 2.0;
when ('TIM_I   ') G_SERVICE = 1.0;
when ('TIME    ') G_SERVICE = 2.0;
when ('URH_I   ') G_SERVICE = 2.0;
when ('URP_I   ') G_SERVICE = 2.0;
when ('UUCP    ') G_SERVICE = 1.0;
when ('UUCP_PAT') G_SERVICE = 0.0;
when ('VMNET   ') G_SERVICE = 1.0;
when ('WHOIS   ') G_SERVICE = 1.0;
otherwise _WARN_ = 'U';
end;
_NORM8 = DMNORM(FLAG, 32.0);
```

```
select (_NORM8);
when ('OTH     ') G_FLAG = 3.0;
when ('REJ     ') G_FLAG = 2.0;
when ('RSTO    ') G_FLAG = 2.0;
when ('RSTOS0  ') G_FLAG = 3.0;
when ('RSTR    ') G_FLAG = 3.0;
when ('S0      ') G_FLAG = 0.0;
when ('S1      ') G_FLAG = 3.0;
when ('S2      ') G_FLAG = 3.0;
when ('S3      ') G_FLAG = 3.0;
when ('SF      ') G_FLAG = 1.0;
when ('SH      ') G_FLAG = 3.0;
otherwise _WARN_ = 'U';
end;
_DM_BAD = 0.0;
_1_0 = 0.0;
_1_1 = 0.0;
_1_2 = 0.0;
_1_3 = 0.0;
_1_4 = 0.0;
_1_5 = 0.0;
_1_6 = 0.0;
_1_7 = 0.0;
_1_8 = 0.0;
_1_9 = 0.0;
_1_10 = 0.0;
_1_11 = 0.0;
_1_12 = 0.0;
_1_13 = 0.0;
_1_14 = 0.0;
if MISSING(AOV16_COUNT) then do ;
_1_0 = .;
_1_1 = .;
_1_2 = .;
_1_3 = .;
_1_4 = .;
_1_5 = .;
_1_6 = .;
_1_7 = .;
_1_8 = .;
_1_9 = .;
_1_10 = .;
_1_11 = .;
_1_12 = .;
_1_13 = .;
_1_14 = .;
substr(_WARN_, 1.0, 1.0) = 'M';
_DM_BAD = 1.0;
end;
 else do ;
_DM12 = put(AOV16_COUNT, BEST12.);
_DM12 = DMNORM(_DM12, 32.0);
_DM_FIND = 0.0;
if _DM12 <= '16' then do ;
if _DM12 <= '12' then do ;
if _DM12 <= '10' then do ;
```

```
if _DM12 = '1' then do ;
_1_0 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '10' then do ;
_1_9 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
 else do ;
if _DM12 = '11' then do ;
_1_10 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '12' then do ;
_1_11 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
end;
 else do ;
if _DM12 <= '14' then do ;
if _DM12 = '13' then do ;
_1_12 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '14' then do ;
_1_13 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
 else do ;
if _DM12 = '15' then do ;
_1_14 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '16' then do ;
_1_0 = -1.0;
_1_1 = -1.0;
_1_2 = -1.0;
_1_3 = -1.0;
_1_4 = -1.0;
_1_5 = -1.0;
_1_6 = -1.0;
_1_7 = -1.0;
_1_8 = -1.0;
_1_9 = -1.0;
_1_10 = -1.0;
_1_11 = -1.0;
```

```
_1_12 = -1.0;
_1_13 = -1.0;
_1_14 = -1.0;
_DM_FIND = 1.0;
end;
end;
end;
end;
end;
 else do ;
if _DM12 <= '5' then do ;
if _DM12 <= '3' then do ;
if _DM12 = '2' then do ;
_1_1 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '3' then do ;
_1_2 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
 else do ;
if _DM12 = '4' then do ;
_1_3 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '5' then do ;
_1_4 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
end;
 else do ;
if _DM12 <= '7' then do ;
if _DM12 = '6' then do ;
_1_5 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '7' then do ;
_1_6 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
 else do ;
if _DM12 = '8' then do ;
_1_7 = 1.0;
_DM_FIND = 1.0;
end;
 else do ;
if _DM12 = '9' then do ;
```

```
_1_8 = 1.0;
_DM_FIND = 1.0;
end;
end;
end;
end;
end;
if ^_DM_FIND then do ;
_1_0 = .;
_1_1 = .;
_1_2 = .;
_1_3 = .;
_1_4 = .;
_1_5 = .;
_1_6 = .;
_1_7 = .;
_1_8 = .;
_1_9 = .;
_1_10 = .;
_1_11 = .;
_1_12 = .;
_1_13 = .;
_1_14 = .;
substr(_WARN_, 2.0, 1.0) = 'U';
_DM_BAD = 1.0;
end;
end;
_2_0 = 0.0;
_2_1 = 0.0;
_2_2 = 0.0;
_2_3 = 0.0;
_2_4 = 0.0;
_2_5 = 0.0;
_2_6 = 0.0;
_2_7 = 0.0;
_2_8 = 0.0;
_2_9 = 0.0;
_2_10 = 0.0;
_2_11 = 0.0;
_2_12 = 0.0;
if MISSING(AOV16_DIF_SRVR) then do ;
_2_0 = .;
_2_1 = .;
_2_2 = .;
_2_3 = .;
_2_4 = .;
_2_5 = .;
_2_6 = .;
_2_7 = .;
_2_8 = .;
_2_9 = .;
_2_10 = .;
_2_11 = .;
_2_12 = .;
substr(_WARN_, 1.0, 1.0) = 'M';
_DM_BAD = 1.0;
```

```
end;
 else do ;
_DM12 = put(AOV16_DIF_SRVR, BEST12.);
_DM12 = DMNORM(_DM12, 32.0);
if _DM12 = '1' then do ;
_2_0 = 1.0;
end;
 else if _DM12 = '2' then do ;
_2_1 = 1.0;
end;
 else if _DM12 = '16' then do ;
_2_0 = -1.0;
_2_1 = -1.0;
_2_2 = -1.0;
_2_3 = -1.0;
_2_4 = -1.0;
_2_5 = -1.0;
_2_6 = -1.0;
_2_7 = -1.0;
_2_8 = -1.0;
_2_9 = -1.0;
_2_10 = -1.0;
_2_11 = -1.0;
_2_12 = -1.0;
end;
 else if _DM12 = '11' then do ;
_2_10 = 1.0;
end;
 else if _DM12 = '8' then do ;
_2_7 = 1.0;
end;
 else if _DM12 = '10' then do ;
_2_9 = 1.0;
end;
 else if _DM12 = '3' then do ;
_2_2 = 1.0;
end;
 else if _DM12 = '7' then do ;
_2_6 = 1.0;
end;
 else if _DM12 = '4' then do ;
_2_3 = 1.0;
end;
 else if _DM12 = '9' then do ;
_2_8 = 1.0;
end;
 else if _DM12 = '5' then do ;
_2_4 = 1.0;
end;
 else if _DM12 = '12' then do ;
_2_11 = 1.0;
end;
 else if _DM12 = '6' then do ;
_2_5 = 1.0;
end;
 else if _DM12 = '13' then do ;
```

```
_2_12 = 1.0;
end;
 else do ;
_2_0 = .;
_2_1 = .;
_2_2 = .;
_2_3 = .;
_2_4 = .;
_2_5 = .;
_2_6 = .;
_2_7 = .;
_2_8 = .;
_2_9 = .;
_2_10 = .;
_2_11 = .;
_2_12 = .;
substr(_WARN_, 2.0, 1.0) = 'U';
_DM_BAD = 1.0;
end;
end;
_3_0 = 0.0;
_3_1 = 0.0;
_3_2 = 0.0;
_3_3 = 0.0;
_3_4 = 0.0;
_3_5 = 0.0;
_3_6 = 0.0;
_3_7 = 0.0;
_3_8 = 0.0;
_3_9 = 0.0;
_3_10 = 0.0;
if MISSING(AOV16_HOT) then do ;
_3_0 = .;
_3_1 = .;
_3_2 = .;
_3_3 = .;
_3_4 = .;
_3_5 = .;
_3_6 = .;
_3_7 = .;
_3_8 = .;
_3_9 = .;
_3_10 = .;
substr(_WARN_, 1.0, 1.0) = 'M';
_DM_BAD = 1.0;
end;
 else do ;
_DM12 = put(AOV16_HOT, BEST12.);
_DM12 = DMNORM(_DM12, 32.0);
if _DM12 = '1' then do ;
_3_0 = 1.0;
end;
 else if _DM12 = '2' then do ;
_3_1 = 1.0;
end;
 else if _DM12 = '15' then do ;
```

```
_3_10 = 1.0;
end;
 else if _DM12 = '3' then do ;
_3_2 = 1.0;
end;
 else if _DM12 = '4' then do ;
_3_3 = 1.0;
end;
 else if _DM12 = '11' then do ;
_3_7 = 1.0;
end;
 else if _DM12 = '12' then do ;
_3_8 = 1.0;
end;
 else if _DM12 = '10' then do ;
_3_6 = 1.0;
end;
 else if _DM12 = '8' then do ;
_3_5 = 1.0;
end;
 else if _DM12 = '16' then do ;
_3_0 = -1.0;
_3_1 = -1.0;
_3_2 = -1.0;
_3_3 = -1.0;
_3_4 = -1.0;
_3_5 = -1.0;
_3_6 = -1.0;
_3_7 = -1.0;
_3_8 = -1.0;
_3_9 = -1.0;
_3_10 = -1.0;
end;
 else if _DM12 = '13' then do ;
_3_9 = 1.0;
end;
 else if _DM12 = '7' then do ;
_3_4 = 1.0;
end;
 else do ;
_3_0 = .;
_3_1 = .;
_3_2 = .;
_3_3 = .;
_3_4 = .;
_3_5 = .;
_3_6 = .;
_3_7 = .;
_3_8 = .;
_3_9 = .;
_3_10 = .;
substr(_WARN_, 2.0, 1.0) = 'U';
_DM_BAD = 1.0;
end;
end;
_5_0 = 0.0;
```

```
_5_1 = 0.0;
_5_2 = 0.0;
_5_3 = 0.0;
_5_4 = 0.0;
_5_5 = 0.0;
_5_6 = 0.0;
_5_7 = 0.0;
_5_8 = 0.0;
_5_9 = 0.0;
_5_10 = 0.0;
_5_11 = 0.0;
_5_12 = 0.0;
_5_13 = 0.0;
_5_14 = 0.0;
if MISSING(AOV16_SRV_CNT) then do ;
_5_0 = .;
_5_1 = .;
_5_2 = .;
_5_3 = .;
_5_4 = .;
_5_5 = .;
_5_6 = .;
_5_7 = .;
_5_8 = .;
_5_9 = .;
_5_10 = .;
_5_11 = .;
_5_12 = .;
_5_13 = .;
_5_14 = .;
substr(_WARN_, 1.0, 1.0) = 'M';
_DM_BAD = 1.0;
end;
 else do ;
_DM12 = put(AOV16_SRV_CNT, BEST12.);
_DM12 = DMNORM(_DM12, 32.0);
if _DM12 = '1' then do ;
_5_0 = 1.0;
end;
 else if _DM12 = '16' then do ;
_5_0 = -1.0;
_5_1 = -1.0;
_5_2 = -1.0;
_5_3 = -1.0;
_5_4 = -1.0;
_5_5 = -1.0;
_5_6 = -1.0;
_5_7 = -1.0;
_5_8 = -1.0;
_5_9 = -1.0;
_5_10 = -1.0;
_5_11 = -1.0;
_5_12 = -1.0;
_5_13 = -1.0;
_5_14 = -1.0;
end;
```

```
 else if _DM12 = '2' then do ;
_5_1 = 1.0;
end;
 else if _DM12 = '15' then do ;
_5_14 = 1.0;
end;
 else if _DM12 = '14' then do ;
_5_13 = 1.0;
end;
 else if _DM12 = '3' then do ;
_5_2 = 1.0;
end;
 else if _DM12 = '4' then do ;
_5_3 = 1.0;
end;
 else if _DM12 = '5' then do ;
_5_4 = 1.0;
end;
 else if _DM12 = '6' then do ;
_5_5 = 1.0;
end;
 else if _DM12 = '8' then do ;
_5_7 = 1.0;
end;
 else if _DM12 = '7' then do ;
_5_6 = 1.0;
end;
 else if _DM12 = '9' then do ;
_5_8 = 1.0;
end;
 else if _DM12 = '10' then do ;
_5_9 = 1.0;
end;
 else if _DM12 = '12' then do ;
_5_11 = 1.0;
end;
 else if _DM12 = '11' then do ;
_5_10 = 1.0;
end;
 else if _DM12 = '13' then do ;
_5_12 = 1.0;
end;
 else do ;
_5_0 = .;
_5_1 = .;
_5_2 = .;
_5_3 = .;
_5_4 = .;
_5_5 = .;
_5_6 = .;
_5_7 = .;
_5_8 = .;
_5_9 = .;
_5_10 = .;
_5_11 = .;
_5_12 = .;
```

```
_5_13 = .;
_5_14 = .;
substr(_WARN_, 2.0, 1.0) = 'U';
_DM_BAD = 1.0;
end;
end;
if MISSING(G_FLAG) then do ;
_6_0 = .;
_6_1 = .;
_6_2 = .;
substr(_WARN_, 1.0, 1.0) = 'M';
_DM_BAD = 1.0;
end;
 else do ;
_DM12 = put(G_FLAG, BEST12.);
_DM12 = DMNORM(_DM12, 32.0);
if _DM12 = '1' then do ;
_6_0 = 0.0;
_6_1 = 1.0;
_6_2 = 0.0;
end;
 else if _DM12 = '0' then do ;
_6_0 = 1.0;
_6_1 = 0.0;
_6_2 = 0.0;
end;
 else if _DM12 = '2' then do ;
_6_0 = 0.0;
_6_1 = 0.0;
_6_2 = 1.0;
end;
 else if _DM12 = '3' then do ;
_6_0 = -1.0;
_6_1 = -1.0;
_6_2 = -1.0;
end;
 else do ;
_6_0 = .;
_6_1 = .;
_6_2 = .;
substr(_WARN_, 2.0, 1.0) = 'U';
_DM_BAD = 1.0;
end;
end;
if MISSING(G_SERVICE) then do ;
_7_0 = .;
_7_1 = .;
substr(_WARN_, 1.0, 1.0) = 'M';
_DM_BAD = 1.0;
end;
 else do ;
_DM12 = put(G_SERVICE, BEST12.);
_DM12 = DMNORM(_DM12, 32.0);
if _DM12 = '2' then do ;
_7_0 = -1.0;
_7_1 = -1.0;
```

```
end;
 else if _DM12 = '0' then do ;
_7_0 = 1.0;
_7_1 = 0.0;
end;
 else if _DM12 = '1' then do ;
_7_0 = 0.0;
_7_1 = 1.0;
end;
 else do ;
_7_0 = .;
_7_1 = .;
substr(_WARN_, 2.0, 1.0) = 'U';
_DM_BAD = 1.0;
end;
end;
if _DM_BAD > 0.0 then do ;
_P0 = 0.0006798097;
_P1 = 0.0153183775;
_P2 = 0.0558123725;
_P3 = 0.3941083163;
_P4 = 0.534081124;
goto REGDR1;
end;
_LP0 = 0.0;
_LP1 = 0.0;
_LP2 = 0.0;
_LP3 = 0.0;
_TEMP = 1.0;
_LP0 = _LP0 + (8.97309749884509) * _TEMP * _1_0;
_LP1 = _LP1 + (9.475456450304) * _TEMP * _1_0;
_LP2 = _LP2 + (0.08183779939133) * _TEMP * _1_0;
_LP3 = _LP3 + (7.91547642280949) * _TEMP * _1_0;
_LP0 = _LP0 + (-7.09311218652648) * _TEMP * _1_1;
_LP1 = _LP1 + (3.42946756538907) * _TEMP * _1_1;
_LP2 = _LP2 + (-1.63736222687037) * _TEMP * _1_1;
_LP3 = _LP3 + (8.60035492871607) * _TEMP * _1_1;
_LP0 = _LP0 + (16.3315840253036) * _TEMP * _1_2;
_LP1 = _LP1 + (-5.85959164693143) * _TEMP * _1_2;
_LP2 = _LP2 + (-2.53740928241609) * _TEMP * _1_2;
_LP3 = _LP3 + (2.62120809028614) * _TEMP * _1_2;
_LP0 = _LP0 + (-22.5615273556858) * _TEMP * _1_3;
_LP1 = _LP1 + (-5.52330111707437) * _TEMP * _1_3;
_LP2 = _LP2 + (-5.33919133360776) * _TEMP * _1_3;
_LP3 = _LP3 + (0.11884727866076) * _TEMP * _1_3;
_LP0 = _LP0 + (-30.2554906468364) * _TEMP * _1_4;
_LP1 = _LP1 + (0.64526397467362) * _TEMP * _1_4;
_LP2 = _LP2 + (-4.40987507627988) * _TEMP * _1_4;
_LP3 = _LP3 + (-1.46254346452609) * _TEMP * _1_4;
_LP0 = _LP0 + (13.4444067104834) * _TEMP * _1_5;
_LP1 = _LP1 + (-15.4359581659106) * _TEMP * _1_5;
_LP2 = _LP2 + (-3.78315830765155) * _TEMP * _1_5;
_LP3 = _LP3 + (-4.74730533646477) * _TEMP * _1_5;
_LP0 = _LP0 + (5.99426137980241) * _TEMP * _1_6;
_LP1 = _LP1 + (3.34304711000097) * _TEMP * _1_6;
_LP2 = _LP2 + (-4.49993737709991) * _TEMP * _1_6;
```

```
_LP3 = _LP3 + (0.39149662840319) * _TEMP * _1_6;
_LP0 = _LP0 + (8.00404660871621) * _TEMP * _1_7;
_LP1 = _LP1 + (3.87729351931859) * _TEMP * _1_7;
_LP2 = _LP2 + (-5.662863418933) * _TEMP * _1_7;
_LP3 = _LP3 + (0.92431512613497) * _TEMP * _1_7;
_LP0 = _LP0 + (9.73639514490121) * _TEMP * _1_8;
_LP1 = _LP1 + (1.66486268124882) * _TEMP * _1_8;
_LP2 = _LP2 + (-5.34790399310294) * _TEMP * _1_8;
_LP3 = _LP3 + (-0.80452936208339) * _TEMP * _1_8;
_LP0 = _LP0 + (-1.18886754533908) * _TEMP * _1_9;
_LP1 = _LP1 + (0.98108751722337) * _TEMP * _1_9;
_LP2 = _LP2 + (-5.09756573837529) * _TEMP * _1_9;
_LP3 = _LP3 + (0.55035390990751) * _TEMP * _1_9;
_LP0 = _LP0 + (3.33003316374041) * _TEMP * _1_10;
_LP1 = _LP1 + (1.28863079547562) * _TEMP * _1_10;
_LP2 = _LP2 + (4.52005620947533) * _TEMP * _1_10;
_LP3 = _LP3 + (-1.88185495205653) * _TEMP * _1_10;
_LP0 = _LP0 + (-1.23514061750629) * _TEMP * _1_11;
_LP1 = _LP1 + (-0.63165164315095) * _TEMP * _1_11;
_LP2 = _LP2 + (4.47980876228159) * _TEMP * _1_11;
_LP3 = _LP3 + (-2.28762571372038) * _TEMP * _1_11;
_LP0 = _LP0 + (3.45175998109795) * _TEMP * _1_12;
_LP1 = _LP1 + (-0.05911640263949) * _TEMP * _1_12;
_LP2 = _LP2 + (3.7133976012504) * _TEMP * _1_12;
_LP3 = _LP3 + (-3.40533163917284) * _TEMP * _1_12;
_LP0 = _LP0 + (-1.79579379752335) * _TEMP * _1_13;
_LP1 = _LP1 + (-0.66575638518718) * _TEMP * _1_13;
_LP2 = _LP2 + (2.46190197688312) * _TEMP * _1_13;
_LP3 = _LP3 + (-3.86144993561858) * _TEMP * _1_13;
_LP0 = _LP0 + (16.2289623285747) * _TEMP * _1_14;
_LP1 = _LP1 + (3.87844062530087) * _TEMP * _1_14;
_LP2 = _LP2 + (13.5342255495752) * _TEMP * _1_14;
_LP3 = _LP3 + (0.11787447033604) * _TEMP * _1_14;
_TEMP = 1.0;
_LP0 = _LP0 + (4.96178727582277) * _TEMP * _2_0;
_LP1 = _LP1 + (7.19423934264755) * _TEMP * _2_0;
_LP2 = _LP2 + (-2.57814751000107) * _TEMP * _2_0;
_LP3 = _LP3 + (-0.41318251862093) * _TEMP * _2_0;
_LP0 = _LP0 + (2.53606187215301) * _TEMP * _2_1;
_LP1 = _LP1 + (1.02456723195019) * _TEMP * _2_1;
_LP2 = _LP2 + (-3.01518942636817) * _TEMP * _2_1;
_LP3 = _LP3 + (-6.42999803474578) * _TEMP * _2_1;
_LP0 = _LP0 + (-17.0716556901489) * _TEMP * _2_2;
_LP1 = _LP1 + (-2.55836176487159) * _TEMP * _2_2;
_LP2 = _LP2 + (-2.66986765613004) * _TEMP * _2_2;
_LP3 = _LP3 + (-3.77427590976266) * _TEMP * _2_2;
_LP0 = _LP0 + (-11.6228431594003) * _TEMP * _2_3;
_LP1 = _LP1 + (-4.42118648129498) * _TEMP * _2_3;
_LP2 = _LP2 + (-2.41006554535669) * _TEMP * _2_3;
_LP3 = _LP3 + (-2.47998713977501) * _TEMP * _2_3;
_LP0 = _LP0 + (-6.65446334079067) * _TEMP * _2_4;
_LP1 = _LP1 + (-4.21089586391698) * _TEMP * _2_4;
_LP2 = _LP2 + (-2.40850931862971) * _TEMP * _2_4;
_LP3 = _LP3 + (-2.46504190674716) * _TEMP * _2_4;
_LP0 = _LP0 + (-3.17136687316047) * _TEMP * _2_5;
_LP1 = _LP1 + (-1.69998112881134) * _TEMP * _2_5;
```

```
_LP2 = _LP2 + (-2.27711189809608) * _TEMP * _2_5;
_LP3 = _LP3 + (-0.56541361679043) * _TEMP * _2_5;
_LP0 = _LP0 + (0.06485838750697) * _TEMP * _2_6;
_LP1 = _LP1 + (2.083825423476) * _TEMP * _2_6;
_LP2 = _LP2 + (4.31755671819224) * _TEMP * _2_6;
_LP3 = _LP3 + (4.36618153369848) * _TEMP * _2_6;
_LP0 = _LP0 + (-3.15969642288067) * _TEMP * _2_7;
_LP1 = _LP1 + (-3.11663563731206) * _TEMP * _2_7;
_LP2 = _LP2 + (-1.93101189518423) * _TEMP * _2_7;
_LP3 = _LP3 + (-0.66813727595772) * _TEMP * _2_7;
_LP0 = _LP0 + (23.3492198306386) * _TEMP * _2_8;
_LP1 = _LP1 + (15.3692429684277) * _TEMP * _2_8;
_LP2 = _LP2 + (19.6299281653522) * _TEMP * _2_8;
_LP3 = _LP3 + (3.7767067535256) * _TEMP * _2_8;
_LP0 = _LP0 + (0.6888846491937) * _TEMP * _2_9;
_LP1 = _LP1 + (0.26881516596812) * _TEMP * _2_9;
_LP2 = _LP2 + (3.49366097063402) * _TEMP * _2_9;
_LP3 = _LP3 + (4.77521196924485) * _TEMP * _2_9;
_LP0 = _LP0 + (6.93832447370645) * _TEMP * _2_10;
_LP1 = _LP1 + (-14.7995817386477) * _TEMP * _2_10;
_LP2 = _LP2 + (-3.17802481741923) * _TEMP * _2_10;
_LP3 = _LP3 + (-0.75953335528334) * _TEMP * _2_10;
_LP0 = _LP0 + (-5.17421740905568) * _TEMP * _2_11;
_LP1 = _LP1 + (-3.50927803184578) * _TEMP * _2_11;
_LP2 = _LP2 + (-0.93991965967767) * _TEMP * _2_11;
_LP3 = _LP3 + (-0.57578867183536) * _TEMP * _2_11;
_LP0 = _LP0 + (-5.40485675039647) * _TEMP * _2_12;
_LP1 = _LP1 + (-3.43007109867235) * _TEMP * _2_12;
_LP2 = _LP2 + (-11.8686117799293) * _TEMP * _2_12;
_LP3 = _LP3 + (-0.57409656273319) * _TEMP * _2_12;
_TEMP = 1.0;
_LP0 = _LP0 + (42.0263556916437) * _TEMP * _3_0;
_LP1 = _LP1 + (1.55172177304255) * _TEMP * _3_0;
_LP2 = _LP2 + (10.9123737543277) * _TEMP * _3_0;
_LP3 = _LP3 + (2.20643367366059) * _TEMP * _3_0;
_LP0 = _LP0 + (35.8542164366111) * _TEMP * _3_1;
_LP1 = _LP1 + (-7.03832251333459) * _TEMP * _3_1;
_LP2 = _LP2 + (-13.5692536842049) * _TEMP * _3_1;
_LP3 = _LP3 + (-11.6512021486838) * _TEMP * _3_1;
_LP0 = _LP0 + (47.2300160154457) * _TEMP * _3_2;
_LP1 = _LP1 + (5.43079775823532) * _TEMP * _3_2;
_LP2 = _LP2 + (-1.76238042211005) * _TEMP * _3_2;
_LP3 = _LP3 + (2.88051687962657) * _TEMP * _3_2;
_LP0 = _LP0 + (32.2801944616028) * _TEMP * _3_3;
_LP1 = _LP1 + (5.10935792540826) * _TEMP * _3_3;
_LP2 = _LP2 + (2.52744460733309) * _TEMP * _3_3;
_LP3 = _LP3 + (2.95088205442946) * _TEMP * _3_3;
_LP0 = _LP0 + (31.6597113950015) * _TEMP * _3_4;
_LP1 = _LP1 + (-9.07258866978128) * _TEMP * _3_4;
_LP2 = _LP2 + (1.62190948241675) * _TEMP * _3_4;
_LP3 = _LP3 + (2.04551962977074) * _TEMP * _3_4;
_LP0 = _LP0 + (31.6597116255105) * _TEMP * _3_5;
_LP1 = _LP1 + (2.67824698076013) * _TEMP * _3_5;
_LP2 = _LP2 + (1.62190948383178) * _TEMP * _3_5;
_LP3 = _LP3 + (1.19293530007666) * _TEMP * _3_5;
_LP0 = _LP0 + (31.6597116340262) * _TEMP * _3_6;
```

```
_LP1 = _LP1 + (2.54298742758522) * _TEMP * _3_6;
_LP2 = _LP2 + (1.62190948388826) * _TEMP * _3_6;
_LP3 = _LP3 + (1.30825513024406) * _TEMP * _3_6;
_LP0 = _LP0 + (-362.950916427088) * _TEMP * _3_7;
_LP1 = _LP1 + (6.17176825281735) * _TEMP * _3_7;
_LP2 = _LP2 + (2.29729057331607) * _TEMP * _3_7;
_LP3 = _LP3 + (1.72970564346861) * _TEMP * _3_7;
_LP0 = _LP0 + (15.9700734501859) * _TEMP * _3_8;
_LP1 = _LP1 + (-9.54799929498259) * _TEMP * _3_8;
_LP2 = _LP2 + (-9.74287510861865) * _TEMP * _3_8;
_LP3 = _LP3 + (1.95662231341111) * _TEMP * _3_8;
_LP0 = _LP0 + (31.6597115840211) * _TEMP * _3_9;
_LP1 = _LP1 + (-9.0725886711641) * _TEMP * _3_9;
_LP2 = _LP2 + (1.62190948358845) * _TEMP * _3_9;
_LP3 = _LP3 + (2.04551963042141) * _TEMP * _3_9;
_LP0 = _LP0 + (31.291502511214) * _TEMP * _3_10;
_LP1 = _LP1 + (20.319207702854) * _TEMP * _3_10;
_LP2 = _LP2 + (1.22785286240863) * _TEMP * _3_10;
_LP3 = _LP3 + (-8.71070773697709) * _TEMP * _3_10;
_TEMP = 1.0;
_LP0 = _LP0 + (39.0432493014866) * _TEMP * _5_0;
_LP1 = _LP1 + (2.41556930669061) * _TEMP * _5_0;
_LP2 = _LP2 + (10.9819053439207) * _TEMP * _5_0;
_LP3 = _LP3 + (-2.4193090445841) * _TEMP * _5_0;
_LP0 = _LP0 + (26.0525989318919) * _TEMP * _5_1;
_LP1 = _LP1 + (-10.8013995852177) * _TEMP * _5_1;
_LP2 = _LP2 + (7.80802468659326) * _TEMP * _5_1;
_LP3 = _LP3 + (-8.37335359162762) * _TEMP * _5_1;
_LP0 = _LP0 + (-91.7996367657177) * _TEMP * _5_2;
_LP1 = _LP1 + (-7.20941847531768) * _TEMP * _5_2;
_LP2 = _LP2 + (6.37205506985912) * _TEMP * _5_2;
_LP3 = _LP3 + (-4.13523264892108) * _TEMP * _5_2;
_LP0 = _LP0 + (-43.2987854849329) * _TEMP * _5_3;
_LP1 = _LP1 + (9.63628678654799) * _TEMP * _5_3;
_LP2 = _LP2 + (15.2260866612625) * _TEMP * _5_3;
_LP3 = _LP3 + (3.41098536758909) * _TEMP * _5_3;
_LP0 = _LP0 + (-92.5078418147566) * _TEMP * _5_4;
_LP1 = _LP1 + (0.92035946274589) * _TEMP * _5_4;
_LP2 = _LP2 + (14.6028124613418) * _TEMP * _5_4;
_LP3 = _LP3 + (4.74556696940043) * _TEMP * _5_4;
_LP0 = _LP0 + (-169.198537792928) * _TEMP * _5_5;
_LP1 = _LP1 + (17.5135430652249) * _TEMP * _5_5;
_LP2 = _LP2 + (-27.5413368656283) * _TEMP * _5_5;
_LP3 = _LP3 + (5.71011491340335) * _TEMP * _5_5;
_LP0 = _LP0 + (29.0429678675398) * _TEMP * _5_6;
_LP1 = _LP1 + (-4.70698581451379) * _TEMP * _5_6;
_LP2 = _LP2 + (2.19747568966552) * _TEMP * _5_6;
_LP3 = _LP3 + (0.25036394861618) * _TEMP * _5_6;
_LP0 = _LP0 + (27.4220001532713) * _TEMP * _5_7;
_LP1 = _LP1 + (-5.62951270960282) * _TEMP * _5_7;
_LP2 = _LP2 + (2.97946845585617) * _TEMP * _5_7;
_LP3 = _LP3 + (0.07300025078033) * _TEMP * _5_7;
_LP0 = _LP0 + (24.9838671156593) * _TEMP * _5_8;
_LP1 = _LP1 + (-4.23916148505361) * _TEMP * _5_8;
_LP2 = _LP2 + (3.42557523365742) * _TEMP * _5_8;
_LP3 = _LP3 + (1.46388562797025) * _TEMP * _5_8;
```

```
_LP0 = _LP0 + (22.8194752422965) * _TEMP * _5_9;
_LP1 = _LP1 + (-4.25224375283395) * _TEMP * _5_9;
_LP2 = _LP2 + (2.49905210556025) * _TEMP * _5_9;
_LP3 = _LP3 + (-0.01709833699071) * _TEMP * _5_9;
_LP0 = _LP0 + (37.114213383863) * _TEMP * _5_10;
_LP1 = _LP1 + (2.9953971574379) * _TEMP * _5_10;
_LP2 = _LP2 + (-4.63754693643679) * _TEMP * _5_10;
_LP3 = _LP3 + (-4.36468726526216) * _TEMP * _5_10;
_LP0 = _LP0 + (34.2320056651284) * _TEMP * _5_11;
_LP1 = _LP1 + (-2.48152127510367) * _TEMP * _5_11;
_LP2 = _LP2 + (-7.20881969172312) * _TEMP * _5_11;
_LP3 = _LP3 + (2.05199646600986) * _TEMP * _5_11;
_LP0 = _LP0 + (34.1979425371632) * _TEMP * _5_12;
_LP1 = _LP1 + (1.32583179116639) * _TEMP * _5_12;
_LP2 = _LP2 + (-1.94011877303868) * _TEMP * _5_12;
_LP3 = _LP3 + (8.74058490108554) * _TEMP * _5_12;
_LP0 = _LP0 + (39.1512435469843) * _TEMP * _5_13;
_LP1 = _LP1 + (1.88577792759584) * _TEMP * _5_13;
_LP2 = _LP2 + (-1.93386166738385) * _TEMP * _5_13;
_LP3 = _LP3 + (0.84886002004651) * _TEMP * _5_13;
_LP0 = _LP0 + (20.9363766085136) * _TEMP * _5_14;
_LP1 = _LP1 + (-2.0647251475618) * _TEMP * _5_14;
_LP2 = _LP2 + (-13.1892422255085) * _TEMP * _5_14;
_LP3 = _LP3 + (-4.52842188369726) * _TEMP * _5_14;
_TEMP = 1.0;
_LP0 = _LP0 + (1.76663561037174) * _TEMP * _6_0;
_LP1 = _LP1 + (-5.40874215787948) * _TEMP * _6_0;
_LP2 = _LP2 + (-6.87281360284862) * _TEMP * _6_0;
_LP3 = _LP3 + (-6.22229997982126) * _TEMP * _6_0;
_LP0 = _LP0 + (21.8797726373068) * _TEMP * _6_1;
_LP1 = _LP1 + (2.87906958740983) * _TEMP * _6_1;
_LP2 = _LP2 + (1.83666665646742) * _TEMP * _6_1;
_LP3 = _LP3 + (4.13135987011355) * _TEMP * _6_1;
_LP0 = _LP0 + (1.73459041116589) * _TEMP * _6_2;
_LP1 = _LP1 + (-0.75352434519744) * _TEMP * _6_2;
_LP2 = _LP2 + (-0.62400019216188) * _TEMP * _6_2;
_LP3 = _LP3 + (0.53569098310408) * _TEMP * _6_2;
_TEMP = 1.0;
_LP0 = _LP0 + (-3.44927846183227) * _TEMP * _7_0;
_LP1 = _LP1 + (-6.37652016665453) * _TEMP * _7_0;
_LP2 = _LP2 + (-4.25904939215537) * _TEMP * _7_0;
_LP3 = _LP3 + (-4.51685639332432) * _TEMP * _7_0;
_LP0 = _LP0 + (-6.43408008433648) * _TEMP * _7_1;
_LP1 = _LP1 + (-0.80236520705753) * _TEMP * _7_1;
_LP2 = _LP2 + (-0.12922463272966) * _TEMP * _7_1;
_LP3 = _LP3 + (-0.63228249961139) * _TEMP * _7_1;
_LPMAX = 0.0;
_LP0 = -123.067467124716 + _LP0;
if _LPMAX < _LP0 then _LPMAX = _LP0;
_LP1 = -23.6221258810818 + _LP1;
if _LPMAX < _LP1 then _LPMAX = _LP1;
_LP2 = -18.5909979689337 + _LP2;
if _LPMAX < _LP2 then _LPMAX = _LP2;
_LP3 = -6.00322742797283 + _LP3;
if _LPMAX < _LP3 then _LPMAX = _LP3;
_LP0 = EXP(_LP0 - _LPMAX);
```

```
_LP1 = EXP(_LP1 - _LPMAX);
_LP2 = EXP(_LP2 - _LPMAX);
_LP3 = EXP(_LP3 - _LPMAX);
_LPMAX = EXP(-_LPMAX);
_P4 = 1.0 / (_LPMAX + _LP0 + _LP1 + _LP2 + _LP3);
_P0 = _LP0 * _P4;
_P1 = _LP1 * _P4;
_P2 = _LP2 * _P4;
_P3 = _LP3 * _P4;
_P4 = _LPMAX * _P4;
REGDR1: P_ATTACKU2R = _P0;
_MAXP = _P0;
_IY = 1.0;
P_ATTACKR2L = _P1;
if (_P1 - _MAXP > 1E-8) then do ;
_MAXP = _P1;
_IY = 2.0;
end;
P_ATTACKPROBE = _P2;
if (_P2 - _MAXP > 1E-8) then do ;
_MAXP = _P2;
_IY = 3.0;
end;
P_ATTACKNORMAL = _P3;
if (_P3 - _MAXP > 1E-8) then do ;
_MAXP = _P3;
_IY = 4.0;
end;
P_ATTACKDOS = _P4;
if (_P4 - _MAXP > 1E-8) then do ;
_MAXP = _P4;
_IY = 5.0;
end;
I_ATTACK = REGDRF[_IY];
U_ATTACK = REGDRU[_IY];
EM_EVENTPROBABILITY = P_ATTACKU2R;
EM_PROBABILITY = MAX(P_ATTACKU2R, P_ATTACKR2L, P_ATTACKPROBE, P_ATTACKNORMAL,
P_ATTACKDOS);
EM_CLASSIFICATION = I_ATTACK;
_return: ;
end;
 enddata;
```

Example of an Input and Output Variables Scoring File

Here is an example of an input and output variables scoring file. The filename is sasscore_score_io.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<Score>
   <Producer>
      <Name> SAS Enterprise Miner </Name>
```

```
      <Version> 1.0 </Version>
   </Producer>
   <TargetList>
   </TargetList>
   <Input>
      <Variable>
         <Name> COUNT </Name>
         <Type> numeric </Type>
      </Variable>
      <Variable>
         <Name> DIF_SRVR </Name>
         <Type> numeric </Type>
         <Description>
            <![CDATA[diff_srv_rate]]>
         </Description>
      </Variable>
      <Variable>
         <Name> FLAG </Name>
         <Type> character </Type>
      </Variable>
      <Variable>
         <Name> HOT </Name>
         <Type> numeric </Type>
      </Variable>
      <Variable>
         <Name> SAM_SRAT </Name>
         <Type> numeric </Type>
         <Description>
            <![CDATA[same_srv_rate]]>
         </Description>
      </Variable>
      <Variable>
         <Name> SERVICE </Name>
         <Type> character </Type>
      </Variable>
      <Variable>
         <Name> SRV_CNT </Name>
         <Type> numeric </Type>
         <Description>
            <![CDATA[srv_count]]>
         </Description>
      </Variable>
   </Input>
   <Output>
      <Variable>
         <Name> AOV16_COUNT </Name>
         <Type> numeric </Type>
      </Variable>
      <Variable>
         <Name> AOV16_DIF_SRVR </Name>
         <Type> numeric </Type>
      </Variable>
      <Variable>
         <Name> AOV16_HOT </Name>
         <Type> numeric </Type>
      </Variable>
```

```
<Variable>
   <Name> AOV16_SAM_SRAT </Name>
   <Type> numeric </Type>
</Variable>
<Variable>
   <Name> AOV16_SRV_CNT </Name>
   <Type> numeric </Type>
</Variable>
<Variable>
   <Name> EM_CLASSIFICATION </Name>
   <Type> character </Type>
   <Description>
      <![CDATA[Prediction for ATTACK]]>
   </Description>
</Variable>
<Variable>
   <Name> EM_EVENTPROBABILITY </Name>
   <Type> numeric </Type>
   <Description>
      <![CDATA[Probability for level U2R of ATTACK]]>
   </Description>
</Variable>
<Variable>
   <Name> EM_PROBABILITY </Name>
   <Type> numeric </Type>
   <Description>
      <![CDATA[Probability of Classification]]>
   </Description>
</Variable>
<Variable>
   <Name> G_FLAG </Name>
   <Type> numeric </Type>
</Variable>
<Variable>
   <Name> G_SERVICE </Name>
   <Type> numeric </Type>
</Variable>
<Variable>
   <Name> I_ATTACK </Name>
   <Type> character </Type>
   <Description>
      <![CDATA[Into: ATTACK]]>
   </Description>
</Variable>
<Variable>
   <Name> P_ATTACKDOS </Name>
   <Type> numeric </Type>
   <Description>
      <![CDATA[Predicted: ATTACK=dos]]>
   </Description>
</Variable>
<Variable>
   <Name> P_ATTACKNORMAL </Name>
   <Type> numeric </Type>
   <Description>
      <![CDATA[Predicted: ATTACK=normal]]>
```

```
      </Description>
   </Variable>
   <Variable>
      <Name> P_ATTACKPROBE </Name>
      <Type> numeric </Type>
      <Description>
         <![CDATA[Predicted: ATTACK=probe]]>
      </Description>
   </Variable>
   <Variable>
      <Name> P_ATTACKR2L </Name>
      <Type> numeric </Type>
      <Description>
         <![CDATA[Predicted: ATTACK=r2l]]>
      </Description>
   </Variable>
   <Variable>
      <Name> P_ATTACKU2R </Name>
      <Type> numeric </Type>
      <Description>
         <![CDATA[Predicted: ATTACK=u2r]]>
      </Description>
   </Variable>
   <Variable>
      <Name> U_ATTACK </Name>
      <Type> character </Type>
      <Description>
         <![CDATA[Unnormalized Into: ATTACK]]>
      </Description>
   </Variable>
   <Variable>
      <Name> _WARN_ </Name>
      <Type> character </Type>
      <Description>
         <![CDATA[Warnings]]>
      </Description>
   </Variable>
</Output>
<C>
   <Function>
      <Name>
         score
      </Name>
      <ParameterList>
         <Parameter>
            <Array length="7">
               <Type>
                  Parm
               </Type>
               <DataMap>
                  <Element index="0">
                     <Value>
                        <Origin> COUNT </Origin>
                        <Type> double </Type>
                     </Value>
                  </Element>
```

```
            <Element index="1">
               <Value>
                  <Origin> DIF_SRVR </Origin>
                  <Type> double </Type>
               </Value>
            </Element>
            <Element index="2">
               <Value>
                  <Origin> FLAG </Origin>
                  <Array length="33">
                    <Type> char </Type>
                  </Array>
               </Value>
            </Element>
            <Element index="3">
               <Value>
                  <Origin> HOT </Origin>
                  <Type> double </Type>
               </Value>
            </Element>
            <Element index="4">
               <Value>
                  <Origin> SAM_SRAT </Origin>
                  <Type> double </Type>
               </Value>
            </Element>
            <Element index="5">
               <Value>
                  <Origin> SERVICE </Origin>
                  <Array length="33">
                    <Type> char </Type>
                  </Array>
               </Value>
            </Element>
            <Element index="6">
               <Value>
                  <Origin> SRV_CNT </Origin>
                  <Type> double </Type>
               </Value>
            </Element>
         </DataMap>
      </Array>
   </Parameter>

   <Parameter>
      <Array length="18">
         <Type>
            Parm
         </Type>
         <DataMap>
            <Element index="0">
               <Value>
                  <Origin> AOV16_COUNT </Origin>
                  <Type> double </Type>
               </Value>
            </Element>
```

```
<Element index="1">
   <Value>
      <Origin> AOV16_DIF_SRVR </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="2">
   <Value>
      <Origin> AOV16_HOT </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="3">
   <Value>
      <Origin> AOV16_SAM_SRAT </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="4">
   <Value>
      <Origin> AOV16_SRV_CNT </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="5">
   <Value>
      <Origin> EM_CLASSIFICATION </Origin>
      <Array length="33">
        <Type> char </Type>
     </Array>
   </Value>
</Element>
<Element index="6">
   <Value>
      <Origin> EM_EVENTPROBABILITY </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="7">
   <Value>
      <Origin> EM_PROBABILITY </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="8">
   <Value>
      <Origin> G_FLAG </Origin>
      <Type> double </Type>
   </Value>
</Element>
<Element index="9">
   <Value>
      <Origin> G_SERVICE </Origin>
      <Type> double </Type>
   </Value>
</Element>
```

```
      <Element index="10">
         <Value>
            <Origin> I_ATTACK </Origin>
            <Array length="7">
              <Type> char </Type>
           </Array>
         </Value>
      </Element>
      <Element index="11">
         <Value>
            <Origin> P_ATTACKDOS </Origin>
            <Type> double </Type>
         </Value>
      </Element>
      <Element index="12">
         <Value>
            <Origin> P_ATTACKNORMAL </Origin>
            <Type> double </Type>
         </Value>
      </Element>
      <Element index="13">
         <Value>
            <Origin> P_ATTACKPROBE </Origin>
            <Type> double </Type>
         </Value>
      </Element>
      <Element index="14">
         <Value>
            <Origin> P_ATTACKR2L </Origin>
            <Type> double </Type>
         </Value>
      </Element>
      <Element index="15">
         <Value>
            <Origin> P_ATTACKU2R </Origin>
            <Type> double </Type>
         </Value>
      </Element>
      <Element index="16">
         <Value>
            <Origin> U_ATTACK </Origin>
            <Array length="7">
              <Type> char </Type>
           </Array>
         </Value>
      </Element>
      <Element index="17">
         <Value>
            <Origin> _WARN_ </Origin>
            <Array length="5">
              <Type> char </Type>
           </Array>
         </Value>
      </Element>
   </DataMap>
</Array>
```

```
            </Parameter>
         </ParameterList>
      </Function>
   </C>
</Score>
```

Example of a User-Defined Formats Scoring File

Here is an example of a user-defined formats scoring file. The filename is sasscore_score_ufmt.xml.

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet type="text/xsl" href="SUVformats.xsl"?>
<LIBRARY type="EXPORT" version="SUV">
   <HEADER>
      <Provider>SAS Institute Inc.</Provider>
      <Version>9.2</Version>
      <VersionLong>9.02.02M2D09012009</VersionLong>
      <CreationDateTime>2009-12-14T12:47:03</CreationDateTime>
   </HEADER>

   <TABLE name="sasscore_score_ufmt">
      <TABLE-HEADER>
         <Provider>SAS Institute Inc.</Provider>
         <Version>9.2</Version>
         <VersionLong>9.02.02M2D09012009</VersionLong>
         <CreationDateTime>2009-12-14T12:47:03</CreationDateTime>
         <ModifiedDateTime>2009-12-14T12:47:03</ModifiedDateTime>

         <Protection />
         <DataSetType />
         <DataRepresentation />
         <Encoding>utf-8</Encoding>
         <ReleaseCreated />
         <HostCreated />
         <FileName>sasscore_score_ufmt</FileName>

         <Observations />
         <Compression number="1" />
         <Variables number="21" />
      </TABLE-HEADER>

      <COLUMN name="FMTNAME" label="Format name">
         <TYPE>character</TYPE>
         <DATATYPE>string</DATATYPE>
         <LENGTH>32</LENGTH>
         <Offset>32</Offset>
         <SortedBy />
      </COLUMN>

      <COLUMN name="START" label="Starting value for format">
         <TYPE>character</TYPE>
         <DATATYPE>string</DATATYPE>
```

```
   <LENGTH>16</LENGTH>
   <Offset>16</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="END" label="Ending value for format">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>16</LENGTH>
   <Offset>16</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="LABEL" label="Format value label">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>3</LENGTH>
   <Offset>3</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="MIN" label="Minimum length">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
   <LENGTH>3</LENGTH>
   <Offset>3</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="MAX" label="Maximum length">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
   <LENGTH>3</LENGTH>
   <Offset>3</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="DEFAULT" label="Default length">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
   <LENGTH>3</LENGTH>
   <Offset>3</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="LENGTH" label="Format length">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
   <LENGTH>3</LENGTH>
   <Offset>3</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="FUZZ" label="Fuzz value">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
```

```
   <LENGTH>8</LENGTH>
   <Offset>8</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="PREFIX" label="Prefix characters">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>2</LENGTH>
   <Offset>2</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="MULT" label="Multiplier">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
   <LENGTH>8</LENGTH>
   <Offset>8</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="FILL" label="Fill character">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>1</LENGTH>
   <Offset>1</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="NOEDIT" label="Is picture string noedit?">
   <TYPE>numeric</TYPE>
   <DATATYPE>double</DATATYPE>
   <LENGTH>3</LENGTH>
   <Offset>3</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="TYPE" label="Type of format">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>1</LENGTH>
   <Offset>1</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="SEXCL" label="Start exclusion">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>1</LENGTH>
   <Offset>1</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="EEXCL" label="End exclusion">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
```

```
   <LENGTH>1</LENGTH>
   <Offset>1</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="HLO" label="Additional information">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>11</LENGTH>
   <Offset>11</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="DECSEP" label="Decimal separator">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>1</LENGTH>
   <Offset>1</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="DIG3SEP" label="Three-digit separator">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>1</LENGTH>
   <Offset>1</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="DATATYPE" label="Date/time/datetime?">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>8</LENGTH>
   <Offset>8</Offset>
   <SortedBy />
</COLUMN>

<COLUMN name="LANGUAGE" label="Language for date strings">
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>8</LENGTH>
   <Offset>8</Offset>
   <SortedBy />
</COLUMN>

<ROW>
   <FMTNAME missing=" " />
   <START missing=" " />
   <END missing=" " />
   <LABEL missing=" " />
   <MIN missing=" " />
   <MAX missing=" " />
   <DEFAULT missing=" " />
   <LENGTH missing=" " />
   <FUZZ missing=" " />
   <PREFIX missing=" " />
```

```
      <MULT missing=" " />
      <FILL missing=" " />
      <NOEDIT missing=" " />
      <TYPE missing=" " />
      <SEXCL missing=" " />
      <EEXCL missing=" " />
      <HLO missing=" " />
      <DECSEP missing=" " />
      <DIG3SEP missing=" " />
      <DATATYPE missing=" " />
      <LANGUAGE missing=" " />
   </ROW>

   <ROW>
      <DELTA-RECORD key="ABC" />
      <FMTNAME>ABC</FMTNAME>
      <START>1</START>
      <END>1</END>
      <LABEL>yes</LABEL>
      <MIN>1</MIN>
      <MAX>40</MAX>
      <DEFAULT>3</DEFAULT>
      <LENGTH>3</LENGTH>
      <FUZZ>1E-12</FUZZ>
      <PREFIX missing=" " />
      <MULT>0</MULT>
      <FILL missing=" " />
      <NOEDIT>0</NOEDIT>
      <TYPE>N</TYPE>
      <SEXCL>N</SEXCL>
      <EEXCL>N</EEXCL>
      <HLO missing=" " />
      <DECSEP missing=" " />
      <DIG3SEP missing=" " />
      <DATATYPE missing=" " />
      <LANGUAGE missing=" " />
   </ROW>

   <ROW>
      <DELTA-RECORD key="YESNO" />
      <FMTNAME>YESNO</FMTNAME>
      <START>0</START>
      <END>0</END>
      <LABEL>NO</LABEL>
      <MIN>1</MIN>
      <MAX>40</MAX>
      <DEFAULT>3</DEFAULT>
      <LENGTH>3</LENGTH>
      <FUZZ>0</FUZZ>
      <PREFIX missing=" " />
      <MULT>0</MULT>
      <FILL missing=" " />
      <NOEDIT>0</NOEDIT>
      <TYPE>C</TYPE>
      <SEXCL>N</SEXCL>
      <EEXCL>N</EEXCL>
```

```
            <HLO missing=" " />
            <DECSEP missing=" " />
            <DIG3SEP missing=" " />
            <DATATYPE missing=" " />
            <LANGUAGE missing=" " />
         </ROW>

         <ROW>
            <FMTNAME>YESNO</FMTNAME>
            <START>1</START>
            <END>1</END>
            <LABEL>YES</LABEL>
            <MIN>1</MIN>
            <MAX>40</MAX>
            <DEFAULT>3</DEFAULT>
            <LENGTH>3</LENGTH>
            <FUZZ>0</FUZZ>
            <PREFIX missing=" " />
            <MULT>0</MULT>
            <FILL missing=" " />
            <NOEDIT>0</NOEDIT>
            <TYPE>C</TYPE>
            <SEXCL>N</SEXCL>
            <EEXCL>N</EEXCL>
            <HLO missing=" " />
            <DECSEP missing=" " />
            <DIG3SEP missing=" " />
            <DATATYPE missing=" " />
            <LANGUAGE missing=" " />
         </ROW>
      </TABLE>
   </LIBRARY>
```

Index

Special Characters

%INDAC_PUBLISH_MODEL macro
  example 21
  overview 17
  running 18
  syntax 19

A

Aster nCluster database 1

D

data mining models 5
deployed components 3

E

EM_ output variables 17
extension nodes 12

F

fixed variable names 11
Formats Library for Aster nCluster 3

M

model registration
  Score Code Export node compared with SAS Metadata Server 6

N

nodes
  score code created by SAS Enterprise Miner nodes 12
  user-defined 12

O

output, created by Score Code Export node 8
output files 9
output variables
  EM_ 17
  naming convention 10

P

process flow diagrams
  for SAS Scoring Accelerator for Aster nCluster 1
  using Score Code Export node in 6
properties, Score Code Export node 7
publishing process 17
publishing the scoring files
  running the %INDAC_PUBLISH_MODEL macro 18

R

registering models
  Score Code Export node compared with SAS Metadata Server 6
Results window 8

S

SAS_SCORE function
  installation 3
  overview 27
  using 27
SAS Embedded Process 3, 17
SAS Enterprise Miner
  score code created by each node 12
SAS Metadata Server
  compared with registering models with Score Code Export node 6

SAS Scoring Accelerator for Aster nCluster
  how it works 1
  overview 1
  process flow diagram 1
score code
  created by each node of SAS Enterprise Miner 12
Score Code Export node 5
  compared with registering models on SAS Metadata Server 6
  files exported by 5
  output created by 8
  properties 7
  using in process flow diagrams 6
scoring files
  creating 17
  example 29, 49, 56
  viewing 25
scoring function 1, 27

U

user-defined nodes 12

V

variables
  fixed variable names 11
  output variables 10