TECHNICAL PAPER

Packaging and Deployment Guidelines for SAS® Models

Last update: August 2021

Contents

Relevant Products and Releases

- SAS® Fraud Management 6.1

Introduction

SAS Fraud Management models that are written in SAS DATA step code are constructed as “packages” that are swappable modules within SAS OnDemand Decision Engine. Therefore, the term model typically refers to the model package. In addition to performing typical model functions such as creating features and scoring, SAS Fraud Management models are also responsible for resolving Signature keys and updating Signatures.

There are two modes in which the models can be packaged for deployment:

- As a compiled SAS macro catalog
- As a set of flat files containing open SAS DATA step code

This document provides information about how to package and deploy a model using these methods in the SAS OnDemand Decision Engine. It is beyond the scope of this document to discuss the design and analytical aspects of the individual components of the model package.

Naming the Model Package

Model packages that are deployed in SAS Fraud Management are named using a 12-byte convention that consists of an 8-byte model ID <*MODEL_ID*> and a 4-byte model version <*MODEL_VERSION*>.

CARDMDL_0100, **BANKMODL0101**

The 12-byte package name is referred to as the PACKAGE_ID in the rest of this document.

Signature Content ID Version

It is very common for the Signature contents to be changed when models are rebuilt. For example, new elements might be added, or existing elements might be removed. Such changes lead to the structure of the Signature content to change as well. Therefore, the Signature content ID version (CIV) serves as a versioning mechanism for different Signature segments.

CIVs are constructed as a 12-byte string that consists of an 8-byte Signature family name and 4-byte version.

CARD_EG_0100, **BANKCS__0101**, **BENEACCT0200**

SAS Fraud Management relies on a Signature definition file for each CIV to parse the Signatures when they are read from or written to the multi-entity history (MEH) database. Therefore, each model package should be accompanied by Signature definition files corresponding to all Signatures that are used by the model. The following code shows how Signature definition files are constructed:

```
/*
* Name: CARD_EG_0100                                                    A
* Key:  RESOLVED                                                        B
* Entity: 00                                                            C
*/

length z00_datetimes_1 - z00_datetimes_5       8.;                     D
length z00_open_to_buy                         8.;
length z00_merchant_1 - z00_merchants_5        $40.;
.
.
```

A. Indicates the CIV that corresponds to this Signature definition.

B. It is beyond the scope of this document to discuss this parameter. However, it is strongly recommended to use the value RESOLVED. When this parameter is set to RESOLVED, it must be accompanied by a corresponding logic in a RESOLVE macro (discussed later).

C. It is beyond the scope of this document to discuss this parameter. However, except in very advanced cases, Entity corresponds to the segment number. Therefore, it can be set to the Signature segment number.

D. Contains the Signature contents decomposed into the constituent scalar variables defined by a series of length statements.

Model Package Components

A model package contains the following components:

1. SCORE contains the actual model code that includes feature generation and scoring. *This is the only mandatory component in a model package.*

   The SCORE component should set values for the following variables:

   - mdl_model_id – should be set to <*MODEL_ID*>
   - mdl_model_version – should be set to <*MODEL_VERSION*>
   - mdl_score – sets the model score. If a score is not calculated for any reason, then it is a good practice to set it to 0.

2. INIT contains code that needs to be executed when the model loads items such as hash declarations, variable declarations, array definitions, and so on.

3. Z##_<*CIV*>_RESOLVE contains logic for resolving a Signature lookup key or keys.

   - *##* indicates the Signature segment to which the resolve logic corresponds.
   - *CIV* indicates the content ID version that corresponds to the Signature segment to which the resolve logic corresponds.

If the model contains multiple Signature segments, then a separate RESOLVE component should be provided. Include one component for each segment that is used.

4. Z##_<*CIV*>_UPDATE contains logic for updating Signatures. If the model contains multiple Signature segments, then a separate UPDATE component should be provided. Include one component for each segment that is used.

5. RESOLVE_RECODE contains code for any optional preprocessing that is required before executing the RESOLVE components.

6. RECODE contains code for any optional preprocessing that is required before executing the UPDATE components.

7. MACROS contains the definitions for various custom macros that are used by the preceding six components.

8. Hash data sets are SAS data sets that are loaded as lookup data sets by the model. Typically, they are loaded as hash objects within the INIT component.

9. Signature definitions describe the layout of each Signature that is used by the model. This component is the only one that is not directly used by the model, but it is required by SAS Fraud Management for setting up Signature Reads and Writes.

Syntactically, components 1 – 6 are required to be structured to execute within a SAS DATA step. They might not contain any SAS procedures or any code that is not compatible with the SAS DATA step. More details are provided in the following sections.

Note: All components (except SCORE) are optional if Signatures are not used by the model. The RESOLVE_RECODE and RECODE components are optional in all scenarios.

High-Level Process Flow

This section provides a high-level overview of how the different model components work together, as shown in Figure 1. It is assumed that the model uses Signatures and that all the optional components are being used as well.

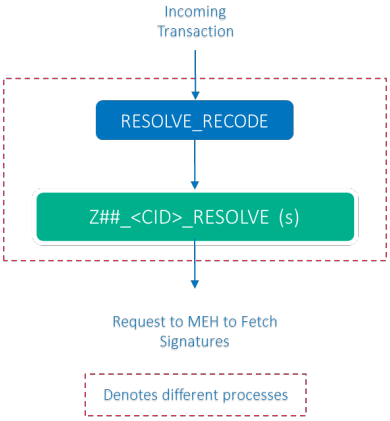

Figure 1. High-Level Process Flow

The incoming transaction is first processed by the RESOLVE_RECODE component. It is next processed by the Signature RESOLVE components. If multiple RESOLVE components exist, then they are processed sequentially. At the completion of this process, the Signature keys are derived, and the system fetches the Signature from the MEH.

The process can be conceptualized by the following DATA step code:

```
data _null_;

  /* Read transaction from an incoming message stream */
  infile stdin;
  input ……;

  %resolve_recode;
  %Z##_<CIV>_RESOLVE;

run;
```

At this stage, the process is cleared by SAS Fraud Management. Therefore, any variables that are created within this DATA step are lost and are not available to the process shown in Figure 2.

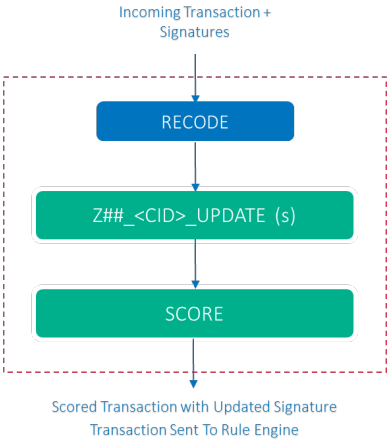

Figure 2. Recode Process

After the Signature is fetched from the MEH, the enriched transaction enters the second process. Here it is first processed by the RECODE component followed by the Signature UPDATE components. If multiple UPDATE components exist, then they are processed sequentially. Finally, the transaction is scored by the SCORE component. In summary, during this process, the Signatures are updated, and the transaction is scored. The scored transaction is passed to the rule engine at this stage. The updated Signature is eventually persisted to the MEH at the end of transaction processing.

The process can be conceptualized by the following DATA step code:

```
data _null_;

  /* Read transaction and Signatures from an incoming message stream */
  infile stdin;
  input ……;

  %recode;
  %Z##_<CIV>_UPDATE;
  %SCORE;

run;
```

Packaging Method 1: Macro Catalog

The model components can be packaged into a single compiled SAS macro catalog. When pursuing this method, the following naming conventions should be followed:

- %MDL_<*PACKAGE_ID*>_SCORE: a macro that encapsulates the SCORE component.
- %MDL_<*PACKAGE_ID*>_INIT: a macro that encapsulates the INIT component.
- %MDL_<*PACKAGE_ID*>_RESOLVE_RECODE: a macro that encapsulates the RESOLVE_RECODE component.
- %MDL_<*PACKAGE_ID*>_RECODE: a macro that encapsulates the RECODE component.
- %Z##_<*CIV*>_RESOLVE: a macro or macros that encapsulate the RESOLVE component or components.
- %Z##_<*CIV*>_UPDATE: a macro or macros that encapsulate the UPDATE component or components.
- Z##_<*CIV*>.SOURCE: a file or files that that contain the Signature definition or definitions.
- Custom macros can be compiled into the catalog without any name changes.

The hash data sets are not included as part of the compiled catalog and are retained as SAS data sets.

The following sample script shows how to compile a SAS macro catalog for package TESTMDL_0100. It assumes that all the individual macros are defined in separate files and are included within the compilation script as shown. It also uses a single Signature on segment 00 with a CIV value of CARDSIG_0100.

```
options sasmstore = compiled mstored;
libname compiled "compiled";

%include "MACROS.sas";                          /* All custom macro definitions */
%include "TESTMDL_0100_INIT.sas";
%include "TESTMDL_0100_RECODE.sas";
%include "TESTMDL_0100_RESOLVE_RECODE.sas";
%include "TESTMDL_0100_SCORE.sas";
%include "Z00_CARD_EG_0100_RESOLVE.sas";
%include "Z00_CARD_EG_0100_UPDATE.sas";

/* Z00_CARDSIG_0100.sas contains the Signature definition */
filename Z00DEF catalog "compiled.SASMACR.Z00_CARD_EG_0100.SOURCE";

data _null_;
  infile "Z00_CARD_EG_0100.sas" lrecl = 80 truncover;
  input ln $char80.;
  file Z00DEF lrecl=80;
  put ln;
run;
```

This program generates the sasmacr.sas7bcat file under the **compiled** directory.

The compiled catalog and the hash data sets constitute the final model package that will be deployed in the SAS OnDemand Decision Engine.

Packaging Method 2: Flat Files

In the second packaging method, the different components can be packaged as separate flat files. When pursuing this method, the following naming conventions should be followed for naming each file:

Note: The file names are case sensitive.

- score.sas
- init.sas
- resolve_recode.sas
- recode.sas
- %Z##_<*CIV*>_resolve.sas
- %Z##_<*CIV*>_update.sas
- Z##DEF: file or files that contain the Signature definition or definitions
- macros.sas: file that contains the definitions of all the custom macros that are used by the model components. For a known issue regarding this file, see Appendix A: Known Issues on page 8.

The hash data sets are retained as SAS data sets.

These individual files and the hash data sets constitute the final model package that will be deployed in the SAS OnDemand Decision Engine.

Model Deployment

To deploy the models for execution on the SAS OnDemand Decision Engine, complete the following steps:

1. Copy the model package to a specified location in SAS Fraud Management.
2. Register the model in the Manager's Workbench.
3. Deploy the model and restart the SAS OnDemand Decision Engine to load the model.

Copy the Model Package

All model packages reside at the location specified by the MODELS_HOME parameter in the ose.properties file. To copy the model package to this location, complete the following steps:

1. Navigate to the location indicated by MODELS_HOME parameter.
2. Create a subdirectory named after the package ID. In the example in this document, it is named **`TESTMDL_0100`**.
3. Copy the compiled macro catalog or the individual flat files to this location.
4. A subdirectory named **`mdlhash`** already exists under MODELS_HOME. Copy all the hash data sets to this location. ***Exercise extreme caution to ensure that data set names do not collide with any existing data sets that are used by existing models.*** If there is a conflict, you might need to rename the data set and change the model code to reflect the new name. It is a good practice to prefix the hash data sets with the package ID or another identifier that is unique to the current model.
5. Set file permissions to make it readable by the OSE user.

Register the Model

For more information, see the section named "Register a Model Package for Use in the SAS OnDemand Decision Engine and Make Real-Time Decisions" in the Model Package Integration Guidelines technical paper that is available from your SAS Implementation Engineer or Technical Support.

Deploy the Model

For more information, see the section named "Deploy the Model Package to the SAS OnDemand Decision Engine" in the Model Package Integration Guidelines technical paper that is available from your SAS Implementation Engineer or Technical Support.

Appendix A: Known Issues

Currently, the system loads the init.sas file before the macros.sas file. This behavior can be problematic if a custom macro that is defined in the macros.sas file is used by the init.sas file. SAS generates an error because it is not able to find this macro definition when executing the init.sas file.

A simple workaround to this problem is to move the contents of the macros.sas file to the beginning of the init.sas file. This workaround enables SAS to load the macro definitions before invoking them from the actual INIT code.

Resources

Note: The following resources are available from your SAS Implementation Engineer or Technical Support.

Model Package Integration Guidelines for SAS® Fraud Management 5

SAS® Fraud Management 6.1: Manager's Workbench User's Guide

Signature Guidelines for SAS® Fraud Management 5

Release Information Content Version: 1.0 August 2021

Trademarks and Patents SAS Institute Inc. SAS Campus Drive, Cary, North Carolina 27513

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. R indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

To contact your local SAS office, please visit: sas.com/offices

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © SAS Institute Inc. All rights reserved.