SAS® Decision Services 5.5

Administrator's Guide

Second Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2012. *SAS® Decision Services 5.5: Administrator's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® Decision Services 5.5: Administrator's Guide, Second Edition

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

August 2012

Contents

What's New

What's New in SAS Decision Services 5.5

Overview

SAS Decision Services (formerly called SAS Real-Time Decision Manager) has the following changes and enhancements:

- Compute tier architecture with significantly improved performance
- DS2 programming language support
- increased I/O throughput with direct database access through JDBC
- support for JBoss servers, in addition to IBM WebSphere Application Server
- SAS BI Web Services support
- Read-only tabular memory cache
- improved load balancing with robust failover and recovery
- rich set of tuning controls
- command-line flow activation and deactivation
- rule sets created with SAS Rule Studio used in decision flows
- database tables can be read from a custom SAS activity

Compute Tier Architecture

SAS Decision Services 5.5 contains significant architectural and performance improvements. Most notable is the replacement of the IBM WebSphere MQ interface to Foundation SAS with JDBC and the DataFlux Federation Server. The DataFlux Federation Server hosts SAS Decision Services activities and score code (predictive models) that are written in the DS2 programming language. It uses SAS threaded-kernel architecture for parallel execution, resulting in higher throughput and lower latency.

DS2 Programming Language Support

SAS activities and score code are now written in the DS2 programming language. DS2 supports a function call syntax that is well suited to real-time programming, and an extended set of data types. The DataFlux Federation Server compiles DS2 code before executing it, resulting in much higher performance. When score code is published to SAS Decision Services by SAS Model Manager, Decision Services converts the code to DS2 and publishes it to the DataFlux Federation Server. SAS Decision Services 5.5 functionality can be extended by writing custom activities in DS2 using the SAS Program Editor.

Increased I/O Throughput

The General I/O activity now uses JDBC for direct access to third-party database management systems, resulting in increased I/O rates. The following database products are supported:

- Oracle
- Microsoft SQL Server

- IBM DB2
- Teradata

SAS data sets are also supported.

Additional Application Servers

The following J2EE application servers are supported by the SAS Decision Services middle tier:

- JBoss
- Oracle WebLogic

IBM WebSphere continues to be supported.

SAS BI Web Services

In addition to the ability to communicate with third-party Web services, special support has been added that enables all SAS Decision Services data types to be exchanged with SAS BI Web Services. This includes all of the integral data types, arrays, and table types. Since multiple DATA step (MDS) activity is no longer supported, BI Web Services enables you to continue to execute SAS procedures or macro code.

Tabular Memory Cache

This new feature allows Read-only tables to be loaded into memory where they can be accessed more quickly than they can when they are read from disk.

Improved Load Balancing and Failover

When hardware failure occurs, SAS Decision Services 5.5 enables optimized process distribution and better performance than earlier releases. Beginning with SAS Decision Services 5.5, every compute tier server is load balanced by every middle-tier server. If a server fails, all remaining servers in the environment continue to process transactions. Recovery is also greatly improved in SAS Decision Services 5.5. When a failed compute tier server comes back online, the system automatically detects it and resumes sending it transactions.

Tuning Controls

JDBC Connection pooling and statement pooling are used to optimize communications, both with database management systems and with the compute tier. Numerous tuning controls are provided to modify the behavior of these pools for optimum performance.

Command-line Flow Activation and Deactivation

The `BatchActivator` command-line utility is an alternative method to using SAS Management Console to activate or deactivate flows to allow scripting.

Publishing Rule Sets as SAS Activities

Rule sets that are created with SAS Rule Studio can now be used in decision flows. Each rule set is expressed as a separate SAS activity.

Reading Database Tables from a Custom SAS Activity

Custom SAS activities can now read records from database tables by using the new "dataset()" method of the DS2 hash object.

Recommended Reading

Here is the recommended reading list for this title:

- *DataFlux Federation Server Administrator’s Guide*
- *SAS 9.3 DS2 Language Reference*
- *SAS 9.3 BI Web Services Developer's Guide*

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

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

1

Overview of SAS Decision Services

What Is SAS Decision Services?

SAS Decision Services combines SAS analytics with business logic to deliver real-time decisions to workflow applications, complex event processors, or interactive customer channels. These channels include the Web, call centers, point of sale (POS) locations, and automated teller machines (ATMs). The product provides an extensible and service-oriented architecture that makes continuous operation possible in environments that have high-transaction volumes.

An administrator performs the following tasks:

- controls which decision flows are in operation at any given time
- promotes decision flows from development to test to production environments
- configures and maintains the SAS Decision Services environments, ensuring that appropriate resources are available within each environment
- monitors the environments in which SAS Decision Services operates

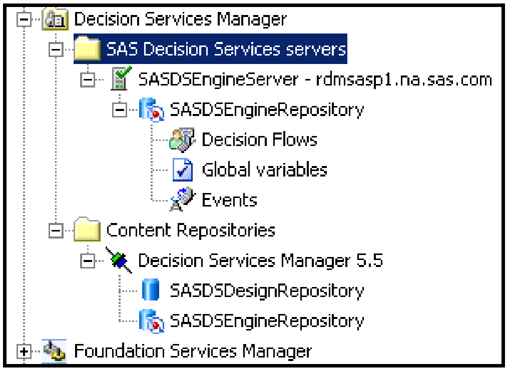
Decision Services Manager Plug-in for SAS Management Console

Most administrative functions are carried out using the Decision Services Manager plug-in. This plug-in is specifically designed for users who want to update, administer, control, or monitor a test or production real-time decision cluster. The plug-in can be used from any client machine that runs SAS Management Console. Users of this plug-in are system administrators, system operators, or performance analysts.

The plug-in is divided into two folders:

- The **`SAS Decision Services servers`** folder provides control of the SAS Decision Services engine server clusters, allowing an administrator to activate and deactivate decision flows and to change the values of global variables.
- The **`Content Repositories`** folder enables an administrator to manage SAS Decision Services repositories and their contents.

Display 1.1 Decision Services Manager Plug-in Folders



```
⊟ Decision Services Manager
   ⊟ SAS Decision Services servers
      ⊟ SASDSEngineServer - rdmsasp1.na.sas.com
         ⊟ SASDSEngineRepository
               Decision Flows
               Global variables
               Events
   ⊟ Content Repositories
      ⊟ Decision Services Manager 5.5
            SASDSDesignRepository
            SASDSEngineRepository
⊞ Foundation Services Manager
```

Icons represent the status of the real-time decision cluster. Here are the icons as well as the type of logical repository that they reference.

Indicates that the plug-in is connected to the SAS Decision Services cluster server.

Indicates that the plug-in cannot connect to the SAS Decision Services cluster server. This typically means that the cluster is not running.

Indicates that the logical repository is a production repository.

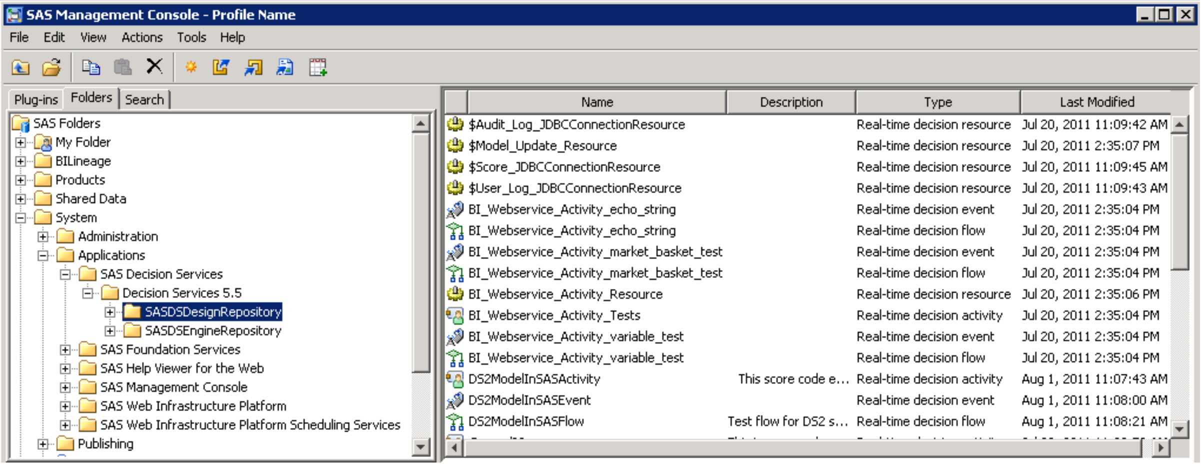
Indicates that the logical repository is a test repository.

Indicates that the logical repository is a development repository.

SAS Decision Services Repository

A SAS Decision Services repository can be viewed in SAS Management Console by using either the **Folders** view or the Decision Services Manager plug-in. In the **Folders** view, each artifact has an associated name, description, type, and modification date.

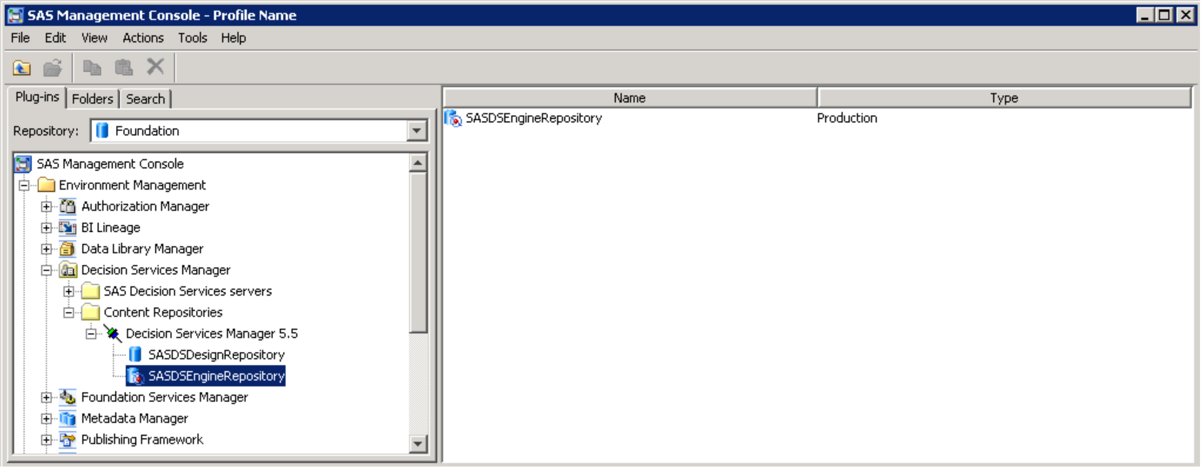
Display 1.2 Decision Services Manager Folders View



Name	Description	Type	Last Modified
$Audit_Log_JDBCConnectionResource		Real-time decision resource	Jul 20, 2011 11:09:42 AM
$Model_Update_Resource		Real-time decision resource	Jul 20, 2011 2:35:07 PM
$Score_JDBCConnectionResource		Real-time decision resource	Jul 20, 2011 11:09:45 AM
$User_Log_JDBCConnectionResource		Real-time decision resource	Jul 20, 2011 11:09:43 AM
BI_Webservice_Activity_echo_string		Real-time decision event	Jul 20, 2011 2:35:04 PM
BI_Webservice_Activity_echo_string		Real-time decision flow	Jul 20, 2011 2:35:04 PM
BI_Webservice_Activity_market_basket_test		Real-time decision event	Jul 20, 2011 2:35:04 PM
BI_Webservice_Activity_market_basket_test		Real-time decision flow	Jul 20, 2011 2:35:04 PM
BI_Webservice_Activity_Resource		Real-time decision resource	Jul 20, 2011 2:35:06 PM
BI_Webservice_Activity_Tests		Real-time decision activity	Jul 20, 2011 2:35:04 PM
BI_Webservice_Activity_variable_test		Real-time decision event	Jul 20, 2011 2:35:04 PM
BI_Webservice_Activity_variable_test		Real-time decision flow	Jul 20, 2011 2:35:04 PM
DS2ModelInSASActivity	This score code e…	Real-time decision activity	Aug 1, 2011 11:07:43 AM
DS2ModelInSASEvent		Real-time decision event	Aug 1, 2011 11:08:00 AM
DS2ModelInSASFlow	Test flow for DS2 s…	Real-time decision flow	Aug 1, 2011 11:08:21 AM

In the Decision Services Manager plug-in, the folder hierarchy is slightly different. It shows a context-sensitive view of the repository and provides product-specific functionality. The **Folders** tab displays a non-context-sensitive view that works with any product. Although rendered differently, both options display the same data.

Display 1.3 Decision Services Manager Plug-ins View



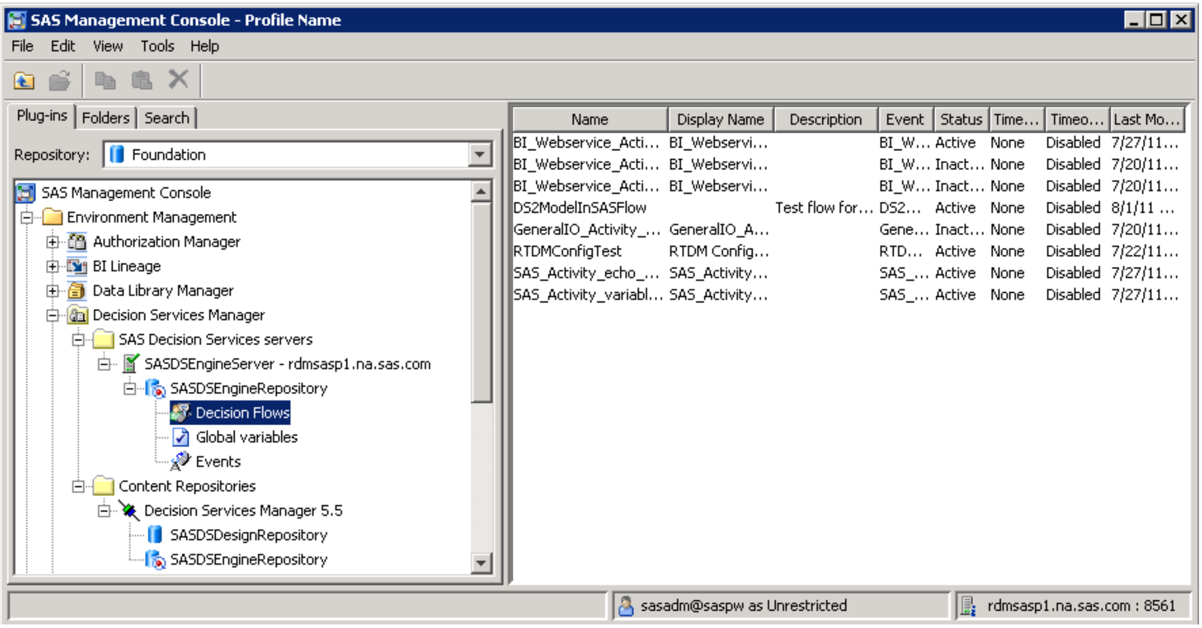
Repository: Foundation

Name	Type
SASDSEngineRepository	Production

When SAS Decision Services artifacts are promoted between development, test, and production environments, the files are copied from one repository to another.

The plug-in displays information about the artifact by reading and interpreting the product-specific metadata. In the following display, **Decision Flows** is selected, and the flow name, display name, description, associated event, status (active or inactive), time-out value, time-out status (enabled or disabled), as well as the last modified date are displayed.

Display 1.4 Decision Flows


SAS Management Console - Profile Name
File Edit View Tools Help
Plug-ins
Folders
Search

Repository: Foundation

- SAS Management Console
  - Environment Management
    - Authorization Manager
    - BI Lineage
    - Data Library Manager
    - Decision Services Manager
      - SAS Decision Services servers
        - SASDSEngineServer - rdmsasp1.na.sas.com
          - SASDSEngineRepository
            - Decision Flows
            - Global variables
            - Events
      - Content Repositories
        - Decision Services Manager 5.5
          - SASDSDesignRepository
          - SASDSEngineRepository

Name	Display Name	Description	Event	Status	Time…	Timeo…	Last Mo…
BI_Webservice_Acti…	BI_Webservi…		BI_W…	Active	None	Disabled	7/27/11…
BI_Webservice_Acti…	BI_Webservi…		BI_W…	Inact…	None	Disabled	7/20/11…
BI_Webservice_Acti…	BI_Webservi…		BI_W…	Inact…	None	Disabled	7/20/11…
DS2ModelInSASFlow		Test flow for…	DS2…	Active	None	Disabled	8/1/11 …
GeneralIO_Activity_…	GeneralIO_A…		Gene…	Inact…	None	Disabled	7/20/11…
RTDMConfigTest	RTDM Config…		RTD…	Active	None	Disabled	7/22/11…
SAS_Activity_echo_…	SAS_Activity…		SAS_…	Active	None	Disabled	7/27/11…
SAS_Activity_variabl…	SAS_Activity…		SAS_…	Active	None	Disabled	7/27/11…

sasadm@saspw as Unrestricted

rdmsasp1.na.sas.com : 8561

2

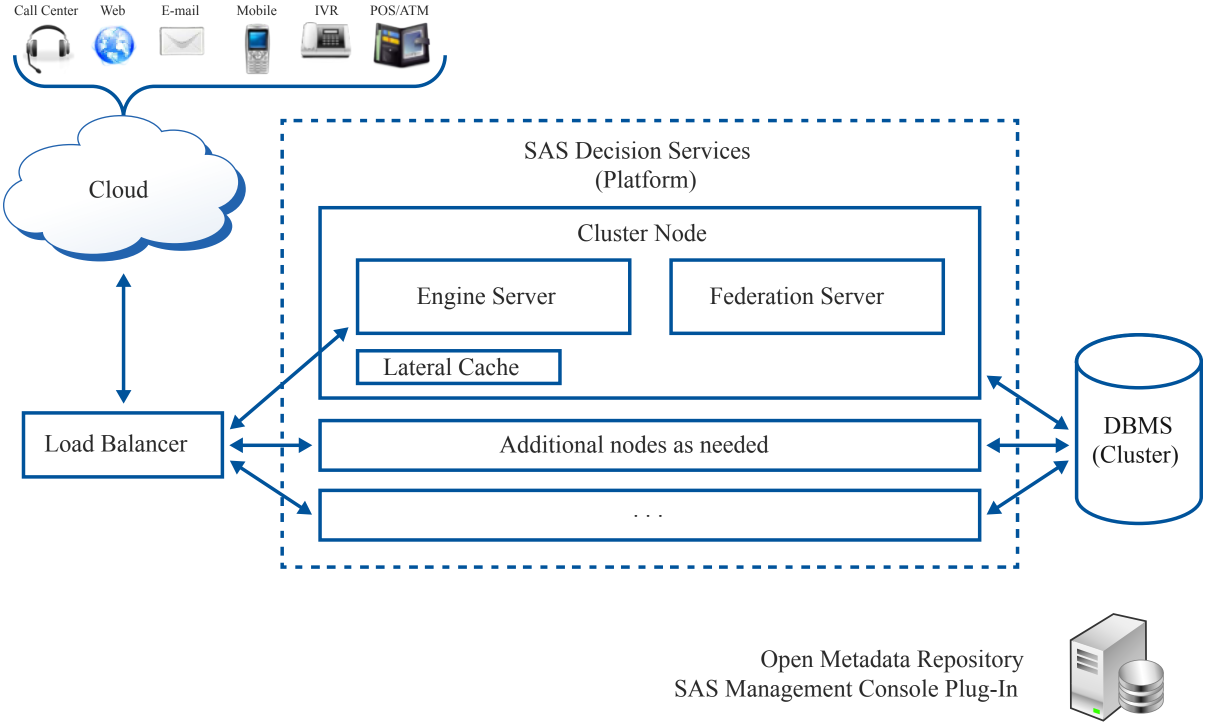
SAS Decision Services Concepts

Scalability and Failover

In SAS Decision Services, horizontal scalability and hardware failover are achieved through server clustering. Vertical scalability and high performance are achieved by the SAS Decision Services engine, which maximizes the parallel processing capabilities of the server hardware. The system is centrally managed using SAS Management Console.

The SAS Decision Services engine is deployed within a Java 2 Enterprise Edition (J2EE) application server container. The clustering and load-balancing capabilities of the application server combine with the SAS Decision Services threaded architecture to enable parallel execution. At any time, servers can be removed from or added to the cluster without stopping the application (for example, if a server fails and restarts). This operation is supported without human intervention: all configuration information that is required to initialize and operate the system is made available in a fail-safe manner within a cluster-wide lateral cache. In addition to the middle tier, SAS Decision Services includes a configurable cluster of SAS servers that bring advanced analytics to the process of making business decisions in a fail-safe manner.

Figure 2.1 *SAS Decision Services Run Time*

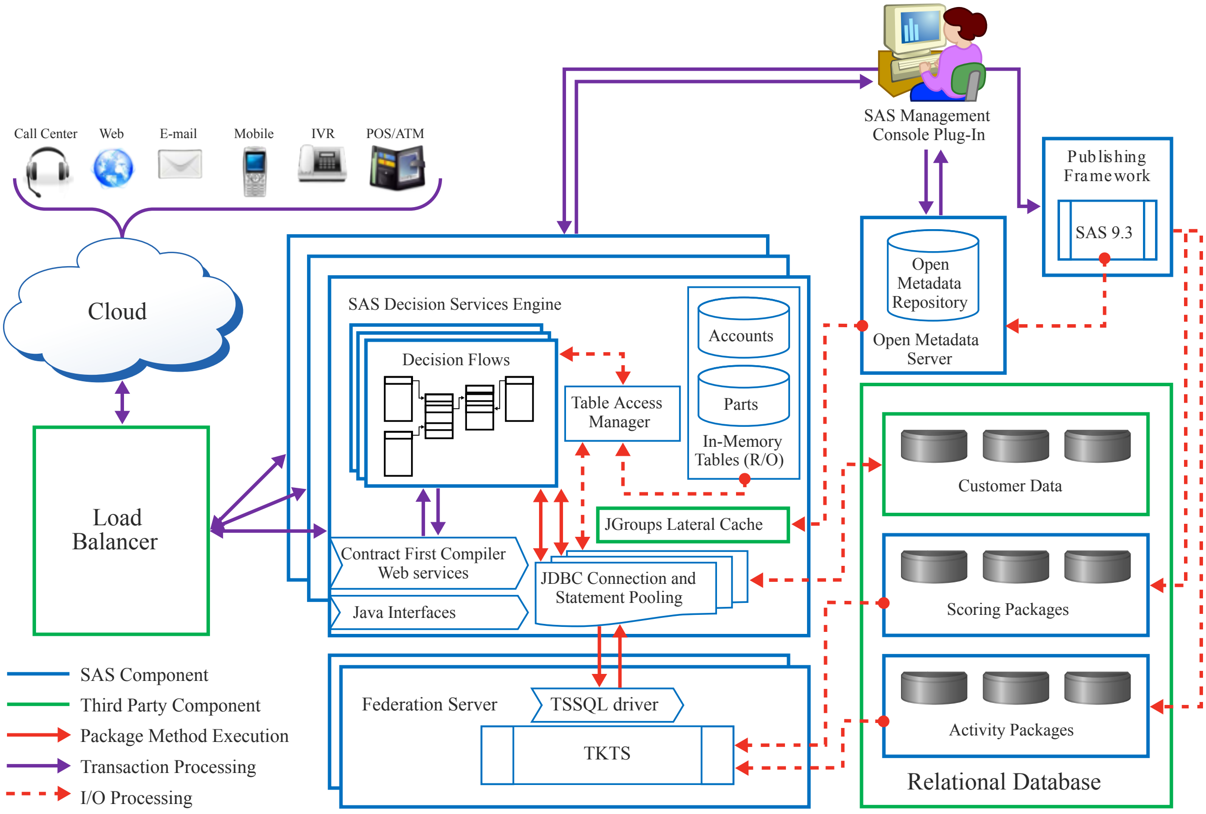


Deployment Topology

Overview

The following figure shows the various logical components of SAS Decision Services when they are deployed in a production environment. The clustering capabilities of this enterprise application provide a highly scalable environment designed to deliver timely real-time analytical decisions.

Figure 2.2 Logical Components in a Production Environment



Production Environment

The production environment consists of either a single instance or multiple instances of the following servers, depending on performance and availability requirements.

- **SAS Metadata Servers** contain artifacts such as global variables, SAS activities, events, and lightweight metadata objects that are pointers to decision flows in the content repository.
- **SAS Decision Services Engine Servers** are configured in an application server cluster. These servers execute the decision flows that provide the real-time analytical decisions.
- **DataFlux Federation Servers** primarily run the SAS activities and score code that are based on DS2.

- **HTTP Servers** are one example of the many available load balancing solutions for the real-time decision cluster enterprise. Using Service-Oriented Architecture (SOA) integration through Web services, HTTP servers can be used as integration points between external applications and a SAS Decision Services cluster.
- **Application Servers** can be configured as a cluster and used for deployment of the SAS Decision Services engine server.
- **Database Servers** store data and DS2 packages, which implement SAS activity methods. SAS servers can be used to run BI Web services for applications that require the execution of procedures or macro code.

The SAS Decision Services cluster enterprise makes extensive use of open standards to simplify integration and maximize interoperability.

A Typical Configuration

A typical installation consists of development, test, and production environments, although the number of environments is configurable to accommodate process standards that reference internal approval. Decision flows are created and functionally tested in the development environment by business users. When a business user is satisfied that a decision flow is ready for deployment, an administrator promotes the flow to either a test or production environment. A test environment is optional and can be used to conduct performance testing on decision flows in an environment that is similar to the production environment. The production environment serves live channels or customer-facing systems. Each environment includes a repository of decision flows, their building blocks, and other resources.

SAS Management Console import and export functionality is used to promote artifacts from one repository to another repository. In this case, decision flows and other artifacts are promoted between development, test, and production environments.

The Decision Services Manager plug-in also operates on these repositories and is used to monitor and control SAS Decision Services run-time systems from a central location.

After a flow is promoted, the Decision Services Manager plug-in can be used to activate the flow, putting it into production.

Development Environment

The development environment enables business users to create, test, edit, and delete decision flows. The SAS Decision Services design server provides this functionality through a Web service API. Client applications provide user-friendly drag-and-drop interfaces, and use the SAS Decision Services design server to execute the above functionality on the users' behalf.

Decision flows and their building blocks (events, activities, global variables, and system resources) are stored in a repository. Each repository resides in SAS Metadata Server. Repositories are managed by the Decision Services Manager plug-in.

A development environment consists of the following components:

- The client application's graphical user interface for building decision flows
- SAS Decision Services Design Server
- A J2EE Application Server
- DataFlux Federation Server and Authentication Server
- SAS Metadata Repository
- SAS Management Console

Test and Production Environments

From a software topology perspective, the test and production environments are identical. The production environment provides the capabilities and performance required for continual operation, twenty-four hours a day, every day of the year.

As with the development environment, decision flows and their building blocks are stored in a repository. Repositories and their contents are managed by the Decision Services Manager plug-in or client application plug-ins. An important function of the Decision Services Manager plug-in (within the test and production environments) is to activate or deactivate decision flows. Activating or deactivating decision flows either connects or disconnects decision flows that have operational channels.

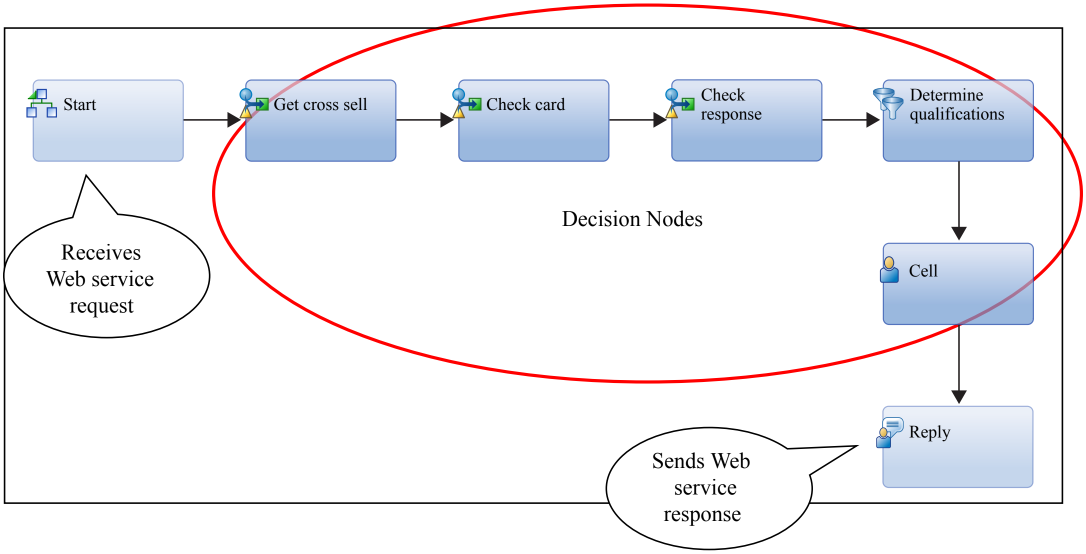
A test or production environment consists of the following components:

- SAS Decision Services engine server cluster and a load balancer
- A J2EE application server (WebSphere, WebLogic, or JBoss) containing the engine server cluster
- One or more DataFlux Federation Servers
- SAS Metadata Repository
- SAS Management Console
- A third-party database management system

Example: The Decision Flow

Consider, for example, a retail business, where SAS Decision Services supports a Web site and an inbound call center. Many decision flows might be deployed to process the various requests that originate from those systems. The following scenario describes a simple example of a cross-sell offer.

Figure 2.3 Scenario - Cross-sell Offer Example



When a customer calls the call center and purchases a product, the customer service representative (CSR) wants to make the best possible cross-sell offer. When the CSR enters the purchase information, the call center application sends a Web service request to SAS Decision Services, requesting the best cross-sell offer to present.

Each active decision flow handles one Web service request type. Therefore, when a cross-sell Web service request is received, the appropriate decision flow processes it. Note that many copies of each decision flow can process multiple requests concurrently and are available to field a high volume of transactions.

In SAS Decision Services, a Web service request is known as an event. Each decision flow begins with a Start activity. When the cross-sell event is received, the Start activity places the relevant request data into a block of in-memory variables known as process variables. In the example, the request data includes the customer's ID and shopping cart items.

The decision flow continues to execute, processing one activity after another, until a Reply activity is reached. The Reply activity sends the results of the decision flow back to the call center via the Web service reply message.

Each activity in a decision flow performs an action. An activity reads the data that is needed to perform its action from the process variables, and it writes the results of that action back to the process variables. In this way, downstream activities can use the outputs of upstream activities as inputs. In the previous example, these are the actions that are performed:

1. Get the request data (Start activity).

2. Retrieve the best cross-sell offer based on the customer’s primary purchase. This step could use any number of SAS analytical techniques, such as scoring the customer with a propensity-to-buy predictive model.

3. Verify that the recommended cross-sell product is not already in the customer’s basket.

4. Check the response history to make sure that the customer has not previously received a cross-sell offer and rejected it.

5 Verify that the customer’s demographic information make her a good candidate for the offer.

6 Record the offer history for future real-time use or offline analysis.

7 Reply with the offer.

More complex decision flows might include branching rules, where the sequence of activity execution is controlled by a set of conditional expressions.

Life Cycle of a Decision Flow

Overview

To deploy a decision flow into production, it must be developed, tested, promoted to a production system, and activated. The following briefly examines each of these stages of the decision flow life cycle. Promotion and activation procedures are described in “Promoting Decision Flows” on page 23.

Development and Testing

Users develop decision flows using the graphical user interface of a client application. This interface allows decision flows to be constructed by dragging and dropping activities from a palette. It also supports the development of decision flow tests.

A significant advantage of the activity model is that business users do not need to understand the complex algorithms used. Rather, they need only to understand how each activity either selects or transforms the data. However, statisticians and other analysts have full access to the underlying algorithms and can change or replace them as needed.

Promotion

SAS Decision Services deployment must include a development environment and a production environment. One or more test environments can be included as well. In this context, a test environment is just like a production environment except that it is not connected to live channels. The type of testing that is performed depends on company policy. Examples include performance testing and verifying flow results over a large set of sample inputs.

When a business user marks a decision flow for deployment, the flow is persisted in a SAS Decision Services repository. If a flow is marked for deployment more than once, then the new copy of the flow overwrites any previous copy. When the flow is persisted, the administrator takes control of the decision flow. The administrator works primarily within SAS Management Console.

Each environment (development, test, and production) has an associated repository. When a user marks a flow for deployment, the client application calls the SAS Decision Services design server that stores the flow in the development repository.

To promote a decision flow, the administrator exports the flow from the development repository and imports it into a test or production repository. (For more information, see “Promoting Decision Flows”.)

Activation

Each decision flow in a test or production environment is either active or inactive. Inactive flows are not loaded by a SAS Decision Services engine server. To put a flow into production (or to make it ready for testing in a test environment) the administrator must activate it. To remove a flow from production, the administrator deactivates it. For more information, see “Activating Flows”.

Decision Flows, Building Blocks, and Artifacts

Overview

A set of activities and system resources is provided with the product and is typically configured by on-site SAS support personnel at the time your system is installed. On-site SAS support personnel can also work with your IT department to define the events that are appropriate to your processing needs. The Decision Services Manager plug-in for SAS Management Console provides advanced functions that support the creation, editing, and deletion of system resources. (For more information, see "Repositories" on page 47.) Other types of artifacts are created or deleted using the SAS Decision Services design server APIs. Client applications use SAS Decision Services design server APIs for this purpose.

Events

Each request for a decision is presented to the system as an event. These events and their associated decision flows are presented to external clients as Web services. An event definition specifies a request message format and a reply message format. Events that are designed only to receive information can omit the reply message. An event makes up the contract between an external system and a decision flow, specifying the types of information contained within the request and reply. Typically, your IT department integrates Web service requests into your systems, and on-site SAS support personnel define the events that make those requests.

Activities

An *activity* is a component of business work such as computing a credit score, or performing a market basket analysis. Activities are represented as the nodes of a decision flow diagram. Each activity contains a set of actions. For example, the general I/O activity contains the actions READ, INSERT, and UPDATE. Each action contains a set of inputs and outputs that are mapped to process variables. The activities that are provided with SAS Decision Services 5.5 contain a rich set of functionality. The activities within a flow can execute sequentially or concurrently as specified by the containing flow.

Decision Flows

A *decision flow* (also called a flow) defines the set of decisions and actions to take when a third-party system, such as a Web site or a call center, sends a request to SAS Decision Services. A decision flow includes activities and business logic that determines the order in which the activities are processed. Each individual type of request has one decision flow that is associated with it. Multiple copies of each decision flow can process multiple requests concurrently and are available to field a high volume of transactions.

Process Variables

Process variables are a set of in-memory typed variables that hold the results of activity actions during flow execution. Process variables enable downstream activities to use the results of upstream activities. For example, a Start activity might write the customer ID that is received from an inbound event to a process variable. Subsequently, a Score activity might be configured to run its Propensity action, which takes the customer ID process variable as input and writes a propensity-to-buy score to another process variable. Following this, the new value of the score might cause a decision activity to branch, and so on.

System Resources

System resources are artifacts that provide activities with access to external resources within their environment, such as relational databases, SAS servers, or Web services. For example, many activities rely on running a SAS DS2 program to produce results. Because flows execute in a J2EE servlet container in the middle tier, these activities must communicate with DataFlux Federation Servers.

The fact that activities reference system resource information (rather than contain system resource information) makes flows portable between systems. SAS Decision Services supports configurable development, test, and production environments. Typically, the set of DataFlux Federation Servers that is used by development and production environments is different. System resources enable the correct set of servers to be used in each environment without modification to the decision flow.

Library Resources

Library resources are special optional system resources that can assist database operations in certain circumstances. Library resources can perform one or both of the following two functions:

- A library resource can be used to specify a list of read-only database tables that are to be read into a memory cache. Access to these tables, through the General I/O activity, is considerably faster than accessing database tables on disk.
- Library resources can hold an alias to a database schema name, allowing the alias name to be used to access tables within the schema. Library resources are optional and are not required for SAS Decision Services operation.

Global Variables

Global variables are used to tune the behavior of flows at execution time. For example, by modifying the value of a global variable that contains a customer risk threshold, the boundary between a medium-risk customer versus a high-risk customer can be adjusted at run time without changing any expressions or redeploying the flow. For more information, see "Managing Global Variables".

Unlike process variables, global variables are read-only with respect to flows and are cluster scoped rather than flow scoped. The value of a global variable affects the behavior of every flow within an engine server cluster that references the global variable.

Sub-flow

A *sub-flow* is a flow that is invoked by another flow. The purpose of sub-flows is to support recursive composition that enables complex flows to be produced by combining simpler, easier-to-understand flows that perform a targeted set of tasks.

There are no distinctions between flows and sub-flows other than the fact that sub-flows are called by other flows. Sub-flows are event-driven like any other flows. To invoke a sub-flow, the user includes a sub-flow activity that enables the user to select the event that drives the desired sub-flow, and to map the event request and reply fields to process variables in the parent flow.

A sub-flow within a particular flow might execute sequentially or concurrently, depending on how the parent flow is configured.

ConcurrentWait Node

This node causes the main flow of execution to wait until all preceding concurrent nodes have finished execution. In case a concurrent node throws an exception, the following ConcurrentWait node captures it and marks it as a fault. The wait in a ConcurrentWait node is timed. If a concurrent node does not complete execution in the given time, the following ConcurrentWait node produces a time-out fault.

If there are no preceding concurrent nodes, then a ConcurrentWait node does not do anything. It is possible to have more than one ConcurrentWait node in a flow. Only those concurrent nodes that are not waited on by a preceding ConcurrentWait node are waited on by the later ConcurrentWait nodes.

Fault Response

Many operations that execute in process-based systems cannot be rolled back (through actions such as sending a message to a third-party system). Therefore, when an error occurs, such systems typically rely on compensation actions rather than on atomic transactions.

Sometimes actions that are performed in real-time, such as sending a message to an operator, cannot be undone. Therefore, when an error occurs, real-time systems typically rely on compensation actions. In cases where a compensation action is not required in the event of an error, a predefined response may be returned to the caller.

Concurrent Execution of Nodes

Activity nodes and sub-flow nodes have an optional Boolean concurrent attribute that indicates whether they should be executed concurrently. If this attribute is true, then these nodes are scheduled for execution on a thread in parallel with the main thread of execution. If the attribute is false, then the nodes execute in sequence. The order of execution of concurrent nodes is indeterminate.

There are three sub-tasks that take place in activity and sub-flow nodes:

1 Process variable values are copied to activity variables or event variables for activity and sub-flow nodes respectively.

2 The actual activity or event is executed.

3 Activity variable or event variable values are copied back to process variable values.

If the nodes are marked concurrent, then step 2 is executed on a separate thread and the main thread continues processing the next node. Step 3, for the concurrent node, is performed when a ConcurrentWait node is reached by the main thread.

There are several implications of this:

1 If there is no ConcurrentWait node following a concurrent node, the output of the concurrent node is not captured as process variable values. Faults and time-outs are also ignored. However, the node does execute. This method could be used for asynchronous execution.

2 The copying back of values to process variables takes place in the main execution thread. However, if the same process variables are referenced for output in other concurrent nodes, the last node is executed.

   Note: The value from the last concurrent node to finish is used. This is indeterminate behavior and is not recommended.

3 In case of an exception, such as a fault or time-out in any concurrent node preceding the ConcurrentWait node, no process variables are updated from that node.

Roles and Capabilities

Overview

SAS Decision Services users are assigned roles that enable them to perform specified actions, or capabilities, in the Decision Services Manager plug-in in SAS Management Console. One or more capabilities can be assigned to a role. For example, the Decision

Services: Advanced role contains capabilities such as viewing content XML, managing repositories, and purging data.

The following roles, with their assigned capabilities, are created during the installation and configuration of SAS Decision Services:

Decision Services: Administration
Provides edit, administrative, and delete capabilities.

Decision Services: Advanced
Provides advanced edit, administrative, and delete capabilities.

Managing Roles and Capabilities

If you have the appropriate permissions, you can create new users and groups and assign roles and capabilities in the SAS Management Console User Manager plug-in. To view or change the capabilities that have been assigned to a role, right-click the role name and select **Properties** ▶ **Capabilities**. SAS Decision Services capabilities are organized into folders. Expand a folder and select a capability to add it to a role.

Best Practices

You can create groups of users and then assign roles to the groups. The best practice is to assign roles to a group, rather than to individual users. You can also create new roles and assign capabilities to them, as well as edit the capabilities of existing roles.

3

Common Operations

Promoting Decision Flows

Overview

You typically promote a flow from a development environment to a test environment, or from a test environment to a production environment. (For more information, see “Life Cycle of a Decision Flow” on page 15.) However, flows and other artifacts can be promoted from any SAS Decision Services repository to any other SAS Decision Services repository. For more information about repositories, see “SAS Decision Services Repository” on page 3.

Promotion Rules

Note: It is highly recommended that you promote only flows and variables.

- **As a general rule, resources should not be promoted.** System resources define how SAS Decision Services interacts with external systems. Because those systems and interactions are different in a production environment than in a development environment, promoting a resource can have undesirable consequences.
- **Do not overwrite an active flow.** If a flow, or other SAS Decision Services object, is promoted to an environment where an object with the same name and type already exists, the object in the target environment can be overwritten. When you overwrite an active flow, the engine is not notified that the flow changed in the repository. Instead of overwriting the flow, deactivate the flow in the target system, promote it, and activate it. These steps cause the engine to load the updated flow. Note that when a flow is promoted, its state is set to inactive.

SAS Decision Services is shipped with a rich set of activities. If your organization develops a new activity that extends SAS Decision Services functionality, that activity must be published to each development, test, or production environment that uses the activity. Any system resources that are referenced by the new activity must also be created in these environments before flows that use the activity are activated.

CAUTION! Activity promotion is not recommended, and can be avoided in most cases. Before promoting any updated activity where a method signature was modified, be sure to deactivate and delete all flows in the target repository that reference the original activity. Failure to do so might yield run-time errors or unexpected results.

When you define a new event (and create a corresponding Web service request that calls SAS Decision Services), then as long as no event with the same name already exists in the target repository, it is safe to promote that event. If you overwrite an existing event, then any active flows or sub-flows that use the event might fail. To update an existing event, make sure that all flows using the original version of the event are deactivated first.

Example: Promotion in SAS Management Console

Promotion is accomplished in SAS Management Console by using the import and export functions from the **Folder** view. Promotion consists of exporting artifacts from one repository and importing them into another repository.

The artifact types that you can export are activity, flow, variable, event, and resource.

CAUTION! The Folder view in SAS Management Console does not restrict the locations to which artifacts can be exported. However, to avoid unpredictable results, always export from an individual artifact.

The following example illustrates the promotion of a flow from a development repository to a production repository. Although both repositories are contained by the same content mapped folder and application context in the example, these conditions are not required.

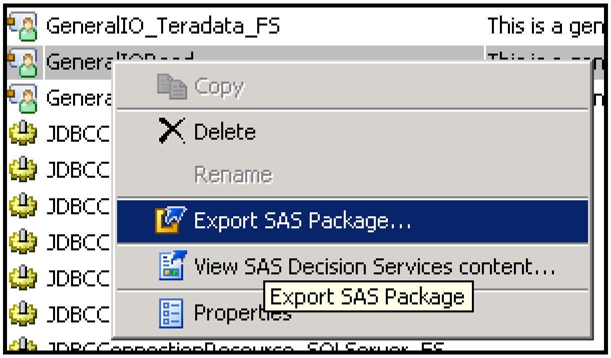
1 Launch SAS Management Console and click the **Folders** tab.

2 Expand the `System` folder and the `Applications` folder.

3 Expand the `SAS Decision Services` and the `Decision Services 5.5` folders.

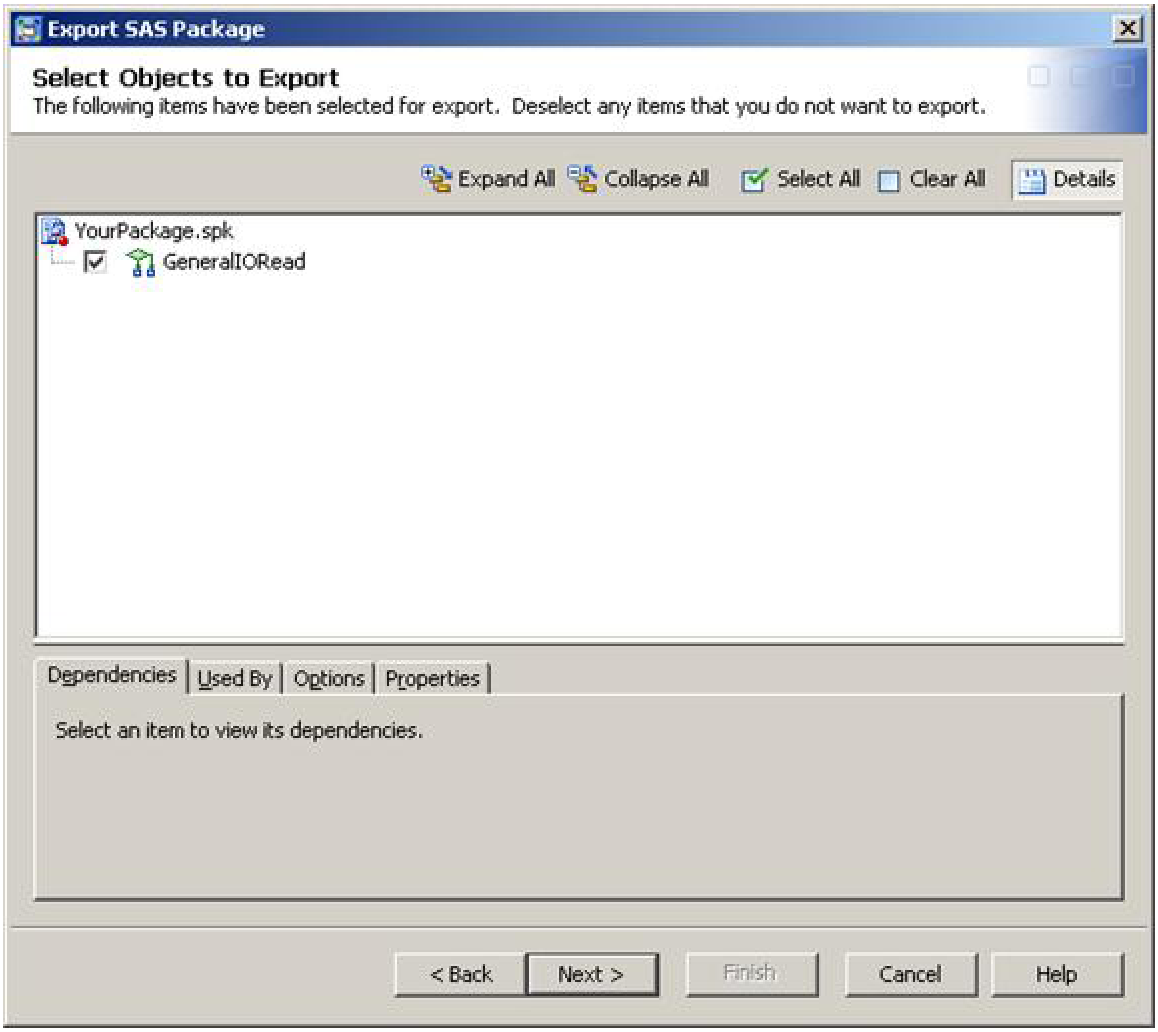
4 Select **SASDSEngineRepository**.

5 Right-click the artifact that you want to promote (for example, GeneralIORead is the artifact shown below), and select **Export SAS Package** (note the previous caution).



6 Enter a package name, and click **Next**. When Export has finished running, the package contains the artifacts that you want to promote.

7 Select the artifacts that you want to promote. A convenient way to select only the boxes that you want is to select **Clear All**. Then select each XML file that you want to promote. Click **Next**.



Select Objects to Export

The following items have been selected for export. Deselect any items that you do not want to export.

YourPackage.spk

☑ GeneralIORead

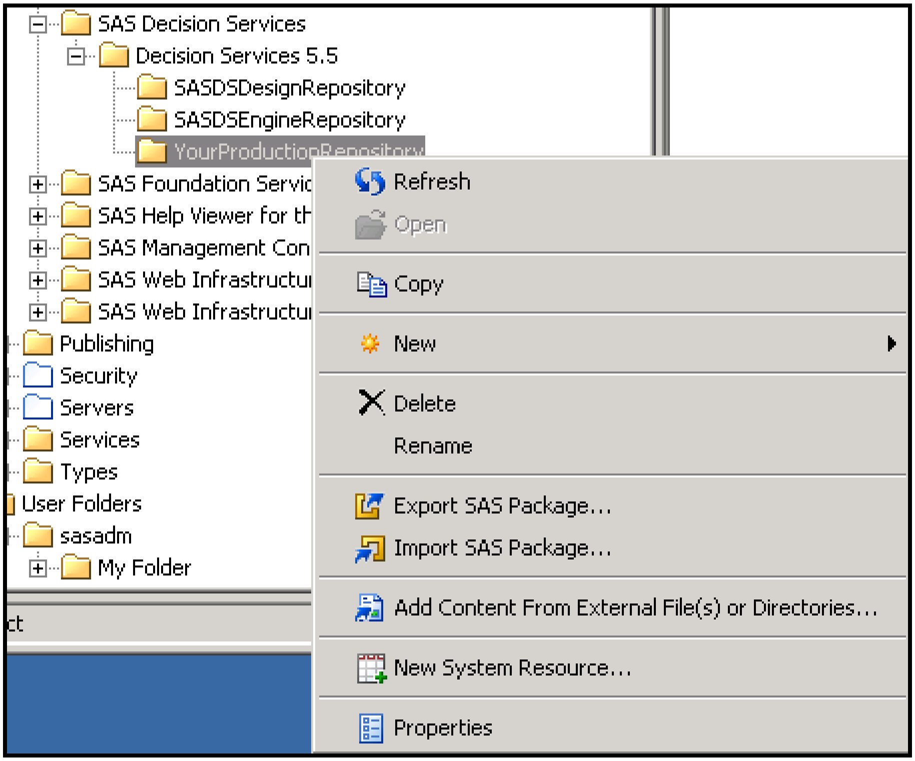
Select an item to view its dependencies.

8 Verify the package name, location, and contents, and click **Export**.

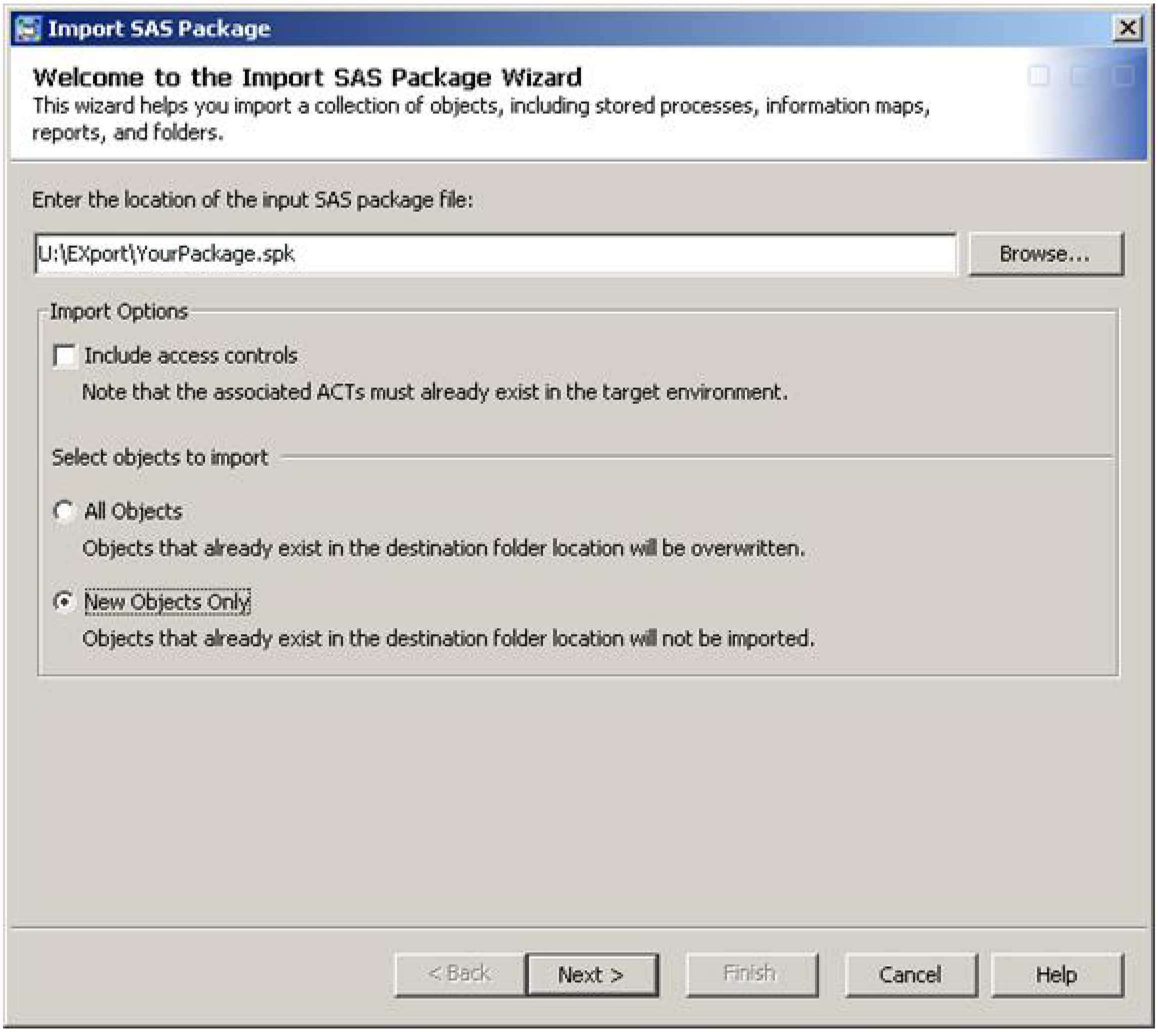
The flow has now been successfully exported from the development environment and saved in the package file called YourPackage.spk. The second part of the promotion process is to import the flow into the production environment.

9 Right-click the repository folder of the repository that you want to promote the artifact to, and select **Import SAS Package**.

CAUTION! The Folder view in SAS Management Console does not restrict the locations to which artifacts can be imported. To avoid unpredictable results, always import to a repository folder.



10 Navigate to your package name. If you import directly after exporting, then the package name is automatically supplied. To avoid overwriting existing artifacts, select **New Objects Only**. Click **Next**.



Import SAS Package

Welcome to the Import SAS Package Wizard

This wizard helps you import a collection of objects, including stored processes, information maps, reports, and folders.

Enter the location of the input SAS package file:

U:\EXport\YourPackage.spk

Import Options

□ Include access controls

Note that the associated ACTs must already exist in the target environment.

Select objects to import

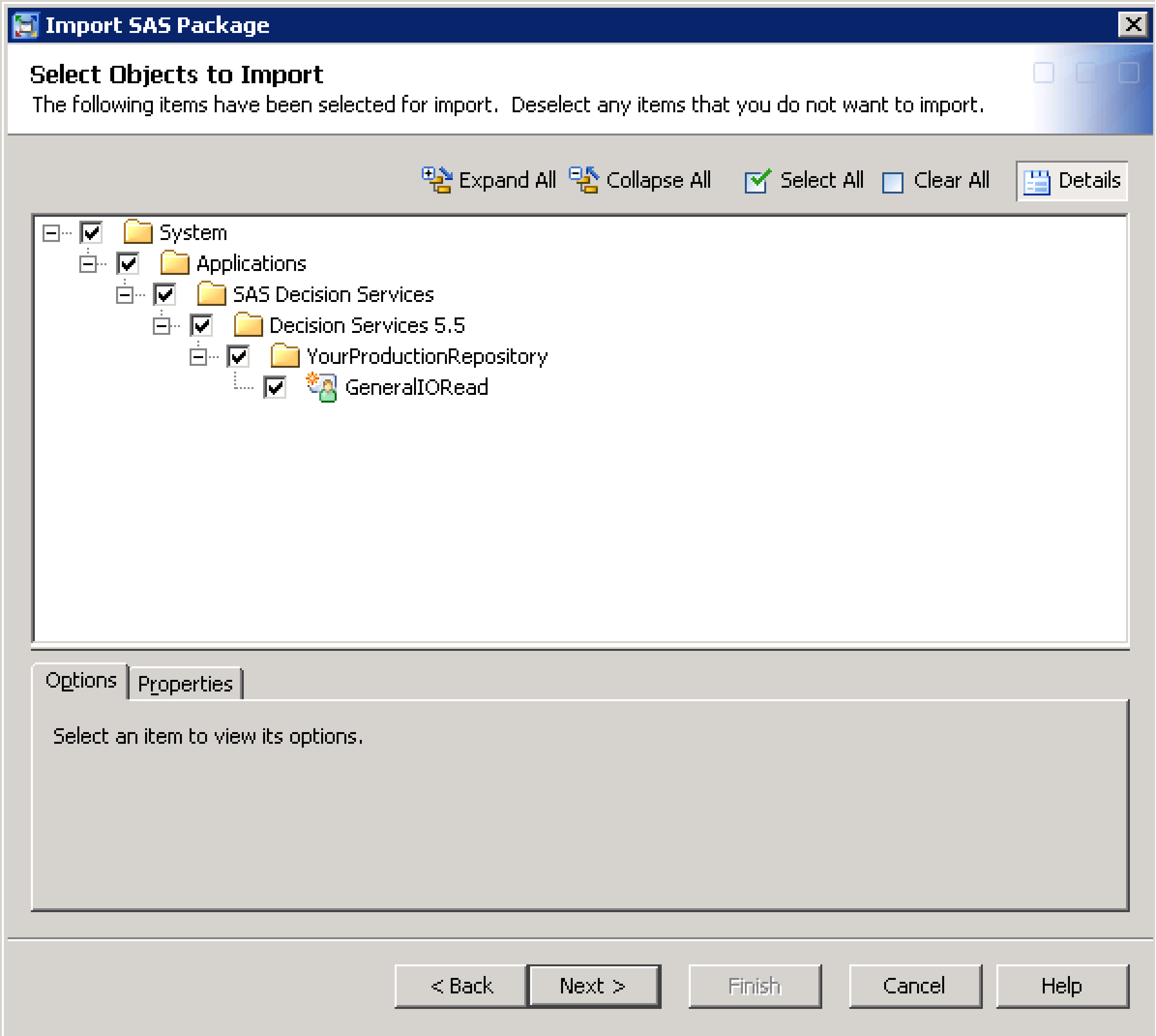
○ All Objects

Objects that already exist in the destination folder location will be overwritten.

● New Objects Only

Objects that already exist in the destination folder location will not be imported.

11 Verify that a check mark exists beside the XML file of each artifact that you selected. Click **Next**.



Import SAS Package

Select Objects to Import

The following items have been selected for import. Deselect any items that you do not want to import.

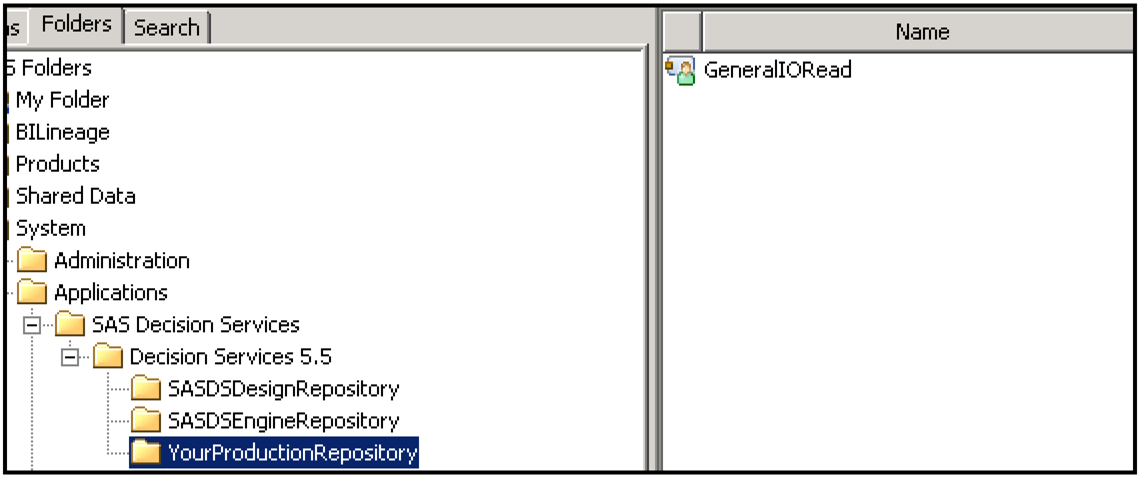
- ☑ System
  - ☑ Applications
    - ☑ SAS Decision Services
      - ☑ Decision Services 5.5
        - ☑ YourProductionRepository
          - ☑ GeneralIORead

Select an item to view its options.

12 Verify that the summary is correct and click **Import**.

13 Click **Finish**.

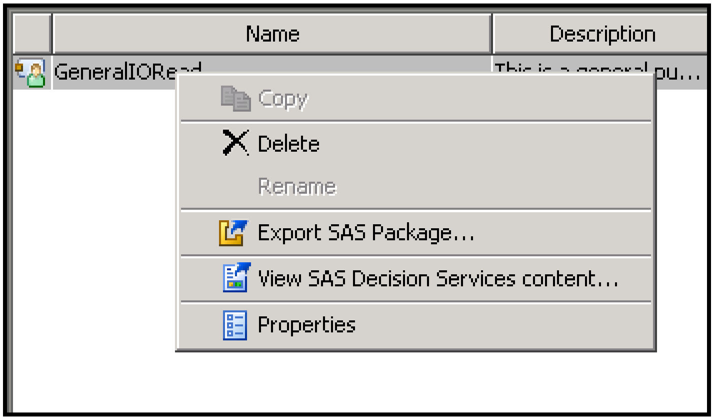
The promotion operation copies the flow without removing the flow from the source repository. The flow has been successfully promoted from the development to the production repositories as shown below.



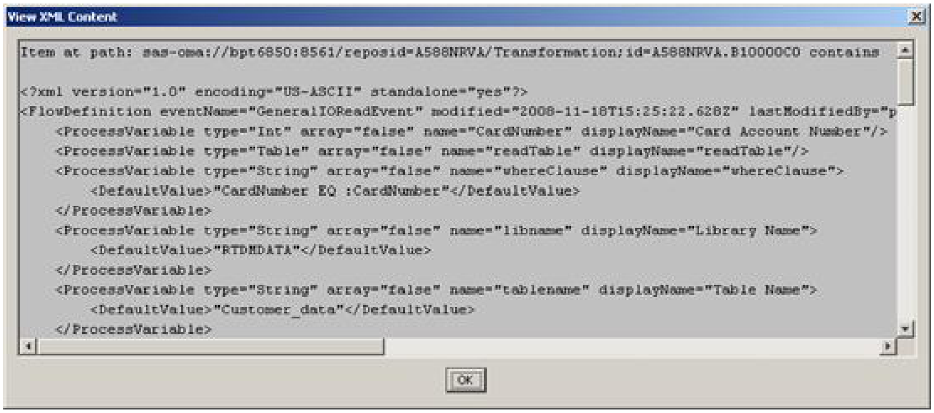
You can further verify that the promotion process was successful by viewing the contents of the XML file after promotion.

1. Click **`YourProductionRepository`** folder so that it appears in the right-hand pane.

2. Right-click **GeneralIORead** and select **View SAS Decision Services content**.



If the XML content can be viewed, then the promotion was successful.



```
Item at path: sas-oma://bpt6850:8561/reposid=A588NRVA/Transformation;id=A588NRVA.B10000C0 contains

<?xml version="1.0" encoding="US-ASCII" standalone="yes"?>
<FlowDefinition eventName="GeneralIOReadEvent" modified="2008-11-18T15:25:22.628Z" lastModifiedBy="p
    <ProcessVariable type="Int" array="false" name="CardNumber" displayName="Card Account Number"/>
    <ProcessVariable type="Table" array="false" name="readTable" displayName="readTable"/>
    <ProcessVariable type="String" array="false" name="whereClause" displayName="whereClause">
        <DefaultValue>"CardNumber EQ :CardNumber"</DefaultValue>
    </ProcessVariable>
    <ProcessVariable type="String" array="false" name="libname" displayName="Library Name">
        <DefaultValue>"RTDMDATA"</DefaultValue>
    </ProcessVariable>
    <ProcessVariable type="String" array="false" name="tablename" displayName="Table Name">
        <DefaultValue>"Customer_data"</DefaultValue>
    </ProcessVariable>
```

Repeat the promotion steps for each artifact type to be promoted.

Activating Flows

When a flow is activated, the engine loads it, making it ready to process events. When a flow is deactivated, the engine unloads it, making it no longer ready to process events. When the engine receives an event for which there is no active flow, it returns a `no flow` message.

A flow is the only artifact that can be activated or deactivated. All other artifacts are used by flows, directly or indirectly, and are loaded when they are referenced by an active flow. When loaded, flows and other artifacts are synchronized across the machines in the SAS Decision Services cluster and cached in memory for maximum performance.

Each flow is bound to an event, which specifies the type of request a flow processes. Many different flows that reference the same event might exist in a repository, but only one of those flows can be active at any given time. For example, suppose flows A and B reference event X, and suppose A is active. Whenever event X is received, it is routed to flow A. If you activate flow B, SAS Decision Services automatically deactivates flow A. Now, whenever event X is received, it is routed to flow B.

It is not necessary to activate or deactivate flows in the development environment. When a flow test is run, SAS Decision Services automatically loads, tests, and unloads the appropriate flow. Because the development environment is not connected to channels, the active or inactive states of the flows there are irrelevant.

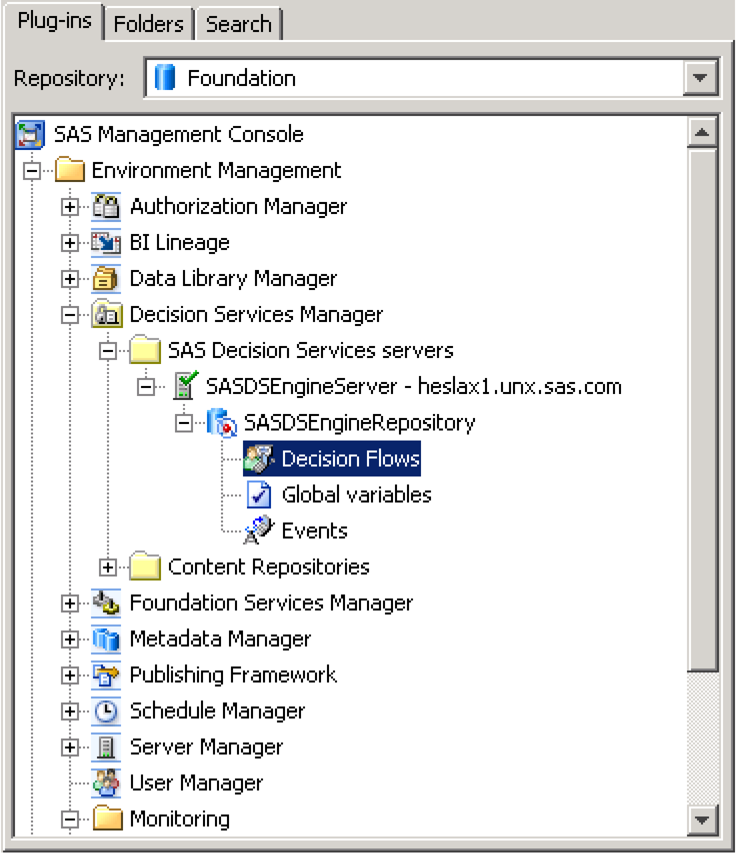
To activate a flow:

1. Launch SAS Management Console.

2. Expand `Decision Services Manager` and the `SAS Decision Services servers` folder.

3. Expand the SAS Decision Services system that contains the flow that you want to activate. In the example below, SASDSEngineServer represents a running engine that is deployed within a cluster. The green check mark indicates that the plug-in has been successfully connected to the engine.

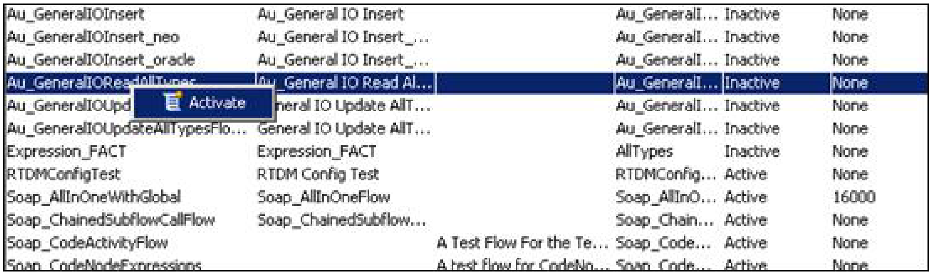
4. Expand the repository (SASDSEngineRepository in the following example) and click **Decision Flows**.



Repository: Foundation

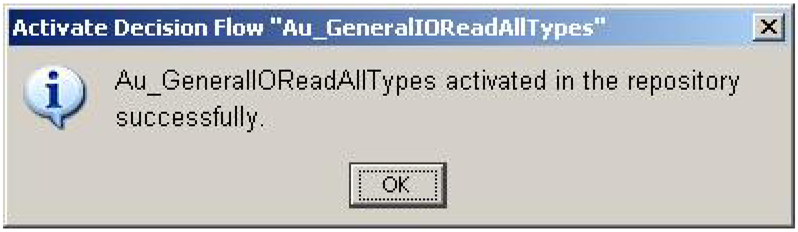
- SAS Management Console
  - Environment Management
    - Authorization Manager
    - BI Lineage
    - Data Library Manager
    - Decision Services Manager
      - SAS Decision Services servers
        - SASDSEngineServer - heslax1.unx.sas.com
          - SASDSEngineRepository
            - Decision Flows
            - Global variables
            - Events
      - Content Repositories
    - Foundation Services Manager
    - Metadata Manager
    - Publishing Framework
    - Schedule Manager
    - Server Manager
    - User Manager
    - Monitoring

5 In the right-hand pane, right-click a flow and select **Activate**.



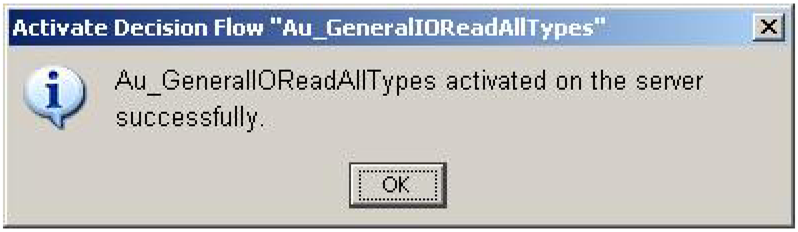
Au_GeneralIOInsert	Au_General IO Insert		Au_GeneralI…	Inactive	None
Au_GeneralIOInsert_neo	Au_General IO Insert_…		Au_GeneralI…	Inactive	None
Au_GeneralIOInsert_oracle	Au_General IO Insert_…		Au_GeneralI…	Inactive	None
Au_GeneralIOReadAllTypes	Au_General IO Read Al…		Au_GeneralI…	Inactive	None
Au_GeneralIOUpd	neral IO Update AllT…		Au_GeneralI…	Inactive	None
Au_GeneralIOUpdateAllTypesFlo…	General IO Update AllT…		Au_GeneralI…	Inactive	None
Expression_FACT	Expression_FACT		AllTypes	Inactive	None
RTDMConfigTest	RTDM Config Test		RTDMConfig…	Active	None
Soap_AllInOneWithGlobal	Soap_AllInOneFlow		Soap_AllInO…	Active	16000
Soap_ChainedSubflowCallFlow	Soap_ChainedSubflow…		Soap_Chain…	Active	None
Soap_CodeActivityFlow		A Test Flow For the Te…	Soap_Code…	Active	None
Soap_CodeNodeExpressions		A test flow for CodeNo…	Soap_Code…	Active	None

When a flow has been successfully activated, the following dialog boxes appear:



Activate Decision Flow "Au_GeneralIOReadAllTypes"

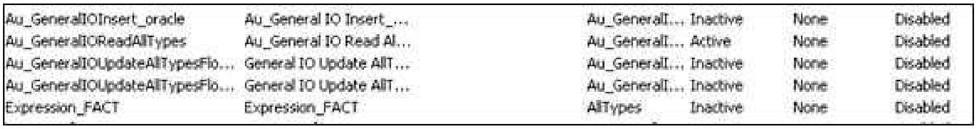
Au_GeneralIOReadAllTypes activated in the repository successfully.



Activate Decision Flow "Au_GeneralIOReadAllTypes"

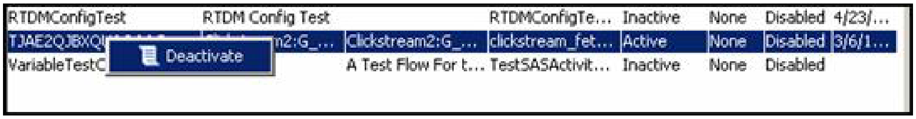
Au_GeneralIOReadAllTypes activated on the server successfully.

The flow status changes from inactive to active, as shown below.



Au_GeneralIOInsert_oracle	Au_General IO Insert_…	Au_GeneralI…	Inactive	None	Disabled
Au_GeneralIOReadAllTypes	Au_General IO Read Al…	Au_GeneralI…	Active	None	Disabled
Au_GeneralIOUpdateAllTypesFlo…	General IO Update AllT…	Au_GeneralI…	Inactive	None	Disabled
Au_GeneralIOUpdateAllTypesFlo…	General IO Update AllT…	Au_GeneralI…	Inactive	None	Disabled
Expression_FACT	Expression_FACT	AllTypes	Inactive	None	Disabled

To deactivate a flow, follow the previous steps in order to view the list of flows. Then right-click an active flow, and select **Deactivate**, as shown below.



RTDMConfigTest	RTDM Config Test		RTDMConfigTe…	Inactive	None	Disabled	4/23/…
TJAE2QJBXQ	m2:G_…	Clickstream2:G_…	clickstream_fet…	Active	None	Disabled	3/6/1…
VariableTestC		A Test Flow For t…	TestSASActivit…	Inactive	None	Disabled	

Flow activation and deactivation can also be scripted, allowing these operations to be controlled by workflow automation software. For a description of the scripting API, see "Activate Flows Using BatchActivator" on page 185.

Managing Global Variables

Global variables are threshold values that are used to tune the behavior of flows at execution time. Unlike process variables that are specific to a flow, the value of a global variable affects the behavior of every flow that references it.

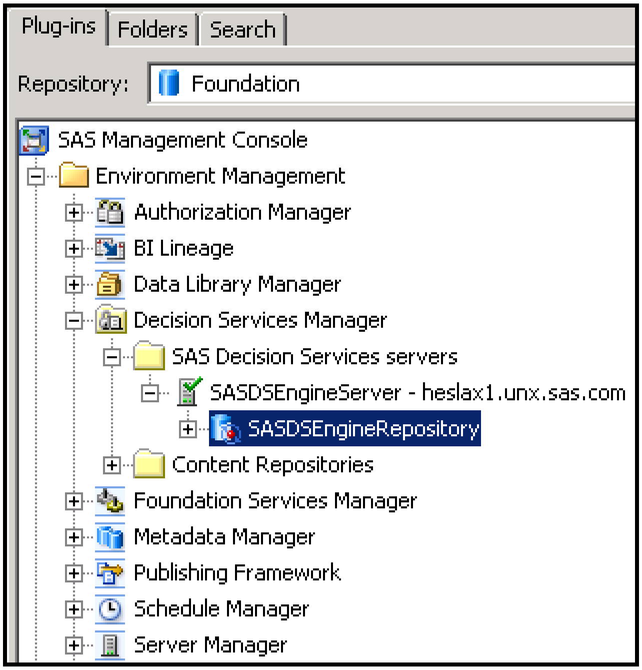
For example, suppose a financial services institution wants to offer premium rates on short-term investment products when more than $10,000 is invested. A global variable called MinimumInvestment with an initial value of `10000.00` might be used in all flows that control the offers of short-term investments. Suppose it is later discovered that money is lost on such investment products when the investment is less than $12,000. Because a global variable was used, its value can easily be adjusted to `12000.00`, rather than modifying every flow that controls the offering of a short-term investment.

Global variables are created and assigned initial values by SAS Customer Intelligence Studio users. For security reasons, users can update only the contents of a development repository. Only an administrator whose role has the Set Global Value capability can change the value of a global variable in a production environment.

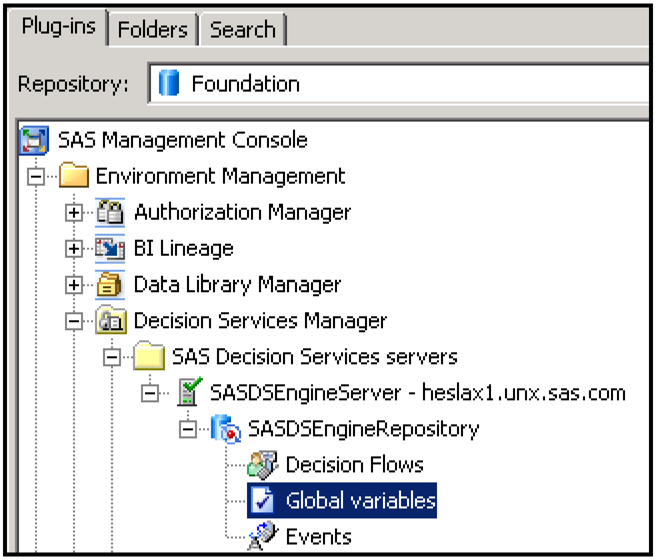
To change the value of a global variable, follow these steps:

1 Launch SAS Management Console.

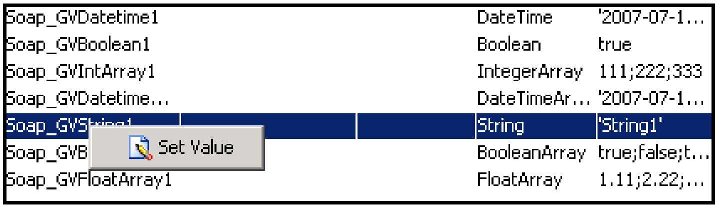
2 On the **Plug-ins** tab, expand **`Decision Services Manager`** and the **`SAS Decision Services servers`** folder.



3 Expand the system that contains the global variable that you want to update. Expand the repository, and select **Global Variables**.

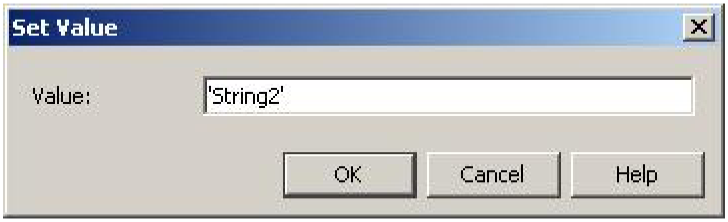


4 Right-click the global variable that you want to change, and click **Set Value**.



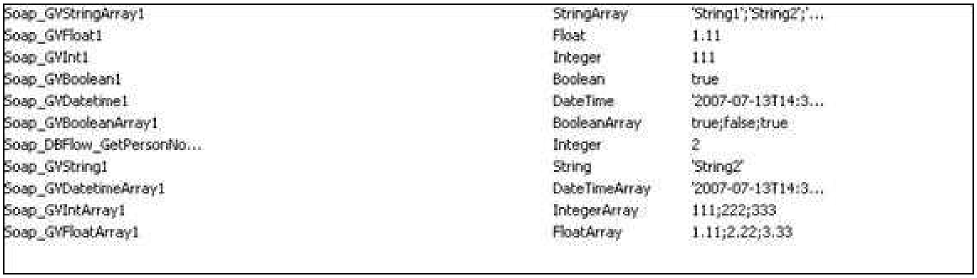
Soap_GVDatetime1	DateTime	'2007-07-1...
Soap_GVBoolean1	Boolean	true
Soap_GVIntArray1	IntegerArray	111;222;333
Soap_GVDatetime...	DateTimeAr...	'2007-07-1...
Soap_GVString1	String	'String1'
Soap_GVB	BooleanArray	true;false;t...
Soap_GVFloatArray1	FloatArray	1.11;2.22;...

5 Type in the new value and click **OK**. Use either single or double quotation marks to indicate a string value.



Value: 'String2'

The new value is displayed in the table on the right pane.



Soap_GVStringArray1	StringArray	'String1';'String2';'...
Soap_GVFloat1	Float	1.11
Soap_GVInt1	Integer	111
Soap_GVBoolean1	Boolean	true
Soap_GVDatetime1	DateTime	'2007-07-13T14:3...
Soap_GVBooleanArray1	BooleanArray	true;false;true
Soap_DBFlow_GetPersonNo...	Integer	2
Soap_GVString1	String	'String2'
Soap_GVDatetimeArray1	DateTimeArray	'2007-07-13T14:3...
Soap_GVIntArray1	IntegerArray	111;222;333
Soap_GVFloatArray1	FloatArray	1.11;2.22;3.33

Set an Event Time-Out

The SAS Decision Services engine receives, processes, and responds to requests that are submitted by the client. When defining an event in SAS Management Console, an administrator is able to specify a time-out setting for the event. It is possible for a decision service that is associated with the event to have a different time-out value setting, even for the same channel. If that is the case, the decision service time-out setting overrides the event time-out setting. This capability ensures that a decision service provides a response within a specified time that is appropriate for the channel and the type of customer interaction. If the run-time engine does not respond to the event request by the time-out interval, fault processing is initiated.

Time-out values can be set at three levels (from lowest to highest): system, event, and flow.

- The system level can be set in the SAS Decision Services enterprise archive file (EAR) during installation and configuration of the design server and engine. If no value is specified by the user, then a default value is set.
- Use the Decision Services Manager plug-in of SAS Management Console to set the time-out value at the event level.
- The flow time-out value supersedes the event and system time-out values. Use the SAS Customer Intelligence Plug-in for SAS Management Console to set the time-out at the flow level. Specify the time-out value interval in milliseconds.

If a sub-flow is called, then the time-out value of the top-level flow or event is used. If the time-out values of the flow or the event are not specified, then the system time-out value is used.

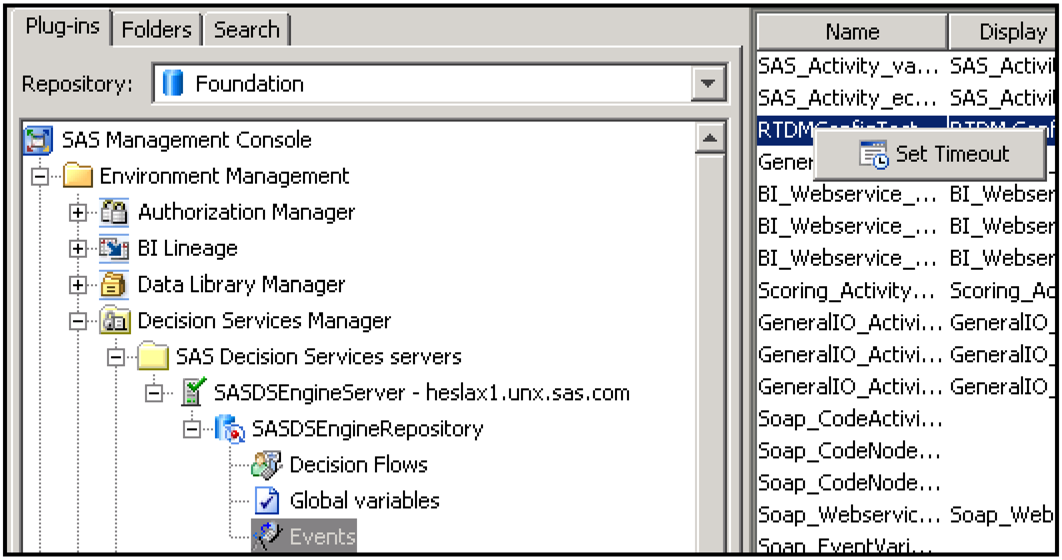
Set the event time-out value by using the Decision Services Manager plug-in for SAS Management Console. To set the time-out value for an event, follow these steps:

1 Launch SAS Management Console.

2 Expand the `Decision Services Manager` and `SAS Decision Services servers` folders.

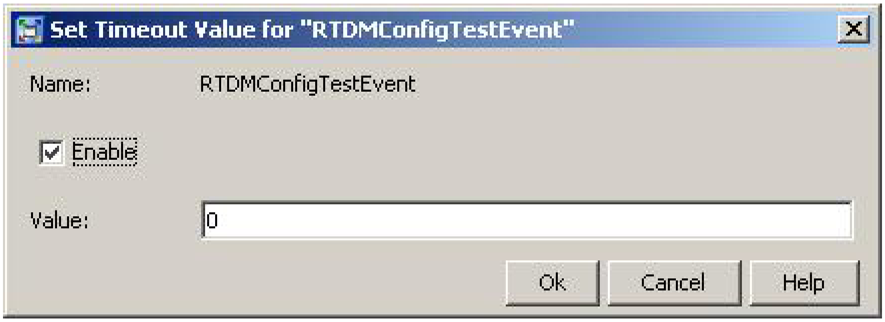
  Expand the system that contains the event that you want to update. Expand the repository and the `Events` folder.

3 Right-click the event that you want to change, and select **Set Timeout**.



Name	Display
SAS_Activity_va...	SAS_Activi
SAS_Activity_ec...	SAS_Activi
RTDM	RTDM
Gener	
BI_Webservice_...	BI_Webser
BI_Webservice_...	BI_Webser
BI_Webservice_...	BI_Webser
Scoring_Activity...	Scoring_Ac
GeneralIO_Activi...	GeneralIO_
GeneralIO_Activi...	GeneralIO_
GeneralIO_Activi...	GeneralIO_
Soap_CodeActivi...	
Soap_CodeNode...	
Soap_CodeNode...	
Soap_Webservic...	Soap_Web
Soap_EventVari...	

4 Select **Enable** to edit the time-out value, enter the value, and click **OK**. If **Enable** is cleared, then the time-out value for the event is disabled.


Set Timeout Value for "RTDMConfigTestEvent"

Name: RTDMConfigTestEvent

☑ Enable

Value: 0

Audit Logging

Audit Logger Overview

The audit logger collects information about events that occur in the SAS Decision Services engine, and records the data in a data table. For more information, see "Tables of Audit Logging Events" on page 42. Events are logged from the SAS Decision Services engine server and from the Decision Services Manager plug-in for SAS Management Console.

These engine events are logged to data tables:

- cached flow
- cached global variables
- engine stop
- flow activate
- flow deactivate

When a global variable value is changed in the Decision Services Manager plug-in, the cached global variables and cached flows are logged.

Terms That Are Used in Audit Logging

The following data items are common to all events:

GUID
: A globally unique ID that is used as the primary key in order to link data in multiple tables.

Host Name
: Used to group several events from an engine server and from the Decision Services Manager plug-in.

Object Name
: A column that contains the name of a flow or of a global variable.

Object Type
: One of the following: an engine, a flow, or a global variable.

Operation
: The type of event that is being logged, such as Cached, Activate, Deactivate, and Stop.

Timestamp
: A sequence of characters that denote the date and time at which a certain event occurred.

Setting Up the Audit Logging Functionality

Overview

The components of the audit logger are configured during the installation and configuration of SAS Decision Services. The $Audit_Log_JDBCConnectionResource is created specifically for the audit logger. It points to the database that is to be used to store the audit log. During configuration, it is possible to set a schema for the audit log. If the schema is left blank, the default schema for the credentials set in the $Audit_Log_JDBCConnectionResource is used.

The data tables (that are required for audit logging) are created during installation or configuration. When SAS Decision Services is installed, no flows are active. Therefore, no entries exist that need to be cached. When a flow is activated, the corresponding event is logged. The engine then caches the flow, and that flow is another event that is logged.

Tables of Audit Logging Events

Audit logging events are recorded in the following four tables. The tables are located in the database that $Audit_Log_JDBCConnectionResource points to.

- AuditLog
- AuditLogFlows
- AuditLogGlobals
- AuditLogGlobalValues

The AuditLogFlows table is related to the AuditLog table via the key GUID. The relationship of AuditLog to AuditLogFlows is one-to-many.

The AuditLogGlobals table is also related to the AuditLog table via the key GUID. The relationship of AuditLog to AuditLogGlobals is one-to-many.

The AuditLogGlobalValues table is related to the AuditLogGlobals table via the key GUID + Name. The relationship of AuditLogGlobals to AuditLogGlobalValues is one-to-many.

Data That Is Logged for Cached Global Variable Events

The following information is logged for each cached global variables event:

- GUID
- Name
- Type
- IsArray
- Value
- Index

Data That Is Logged for Cached Flow Events

The following information is logged for each cached flows event:

- GUID
- Name

- Type

Data Collection for Performance Analysis

The SAS Decision Services User Log

Overview

The user log collects information about how specific events flow through the engine. It should be used for short periods of time only. It can also be used to debug specific events. Do not use the user log in the production environment because it has a significant performance impact.

The JDBC Connection system resource that is used by the user log is specified in the rtdm_config.properties file as rtdm.user.log.resource.name. This value is set as part of the configuration process.

What the SAS Decision Services User Log Contains

The user log contains several XML documents. Each document has a top level element called TestOutput. Each XML document represents one decision flow invocation and contains all of the following information:

- the event request data
- the values of the process variables before executing each activity
- the values of the process variables after executing each activity
- the path that the event traveled through the flow

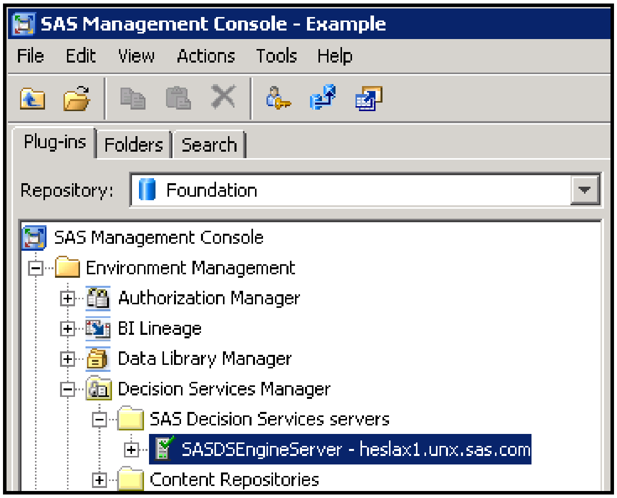
Location of the User Log

The location of the user log is set during the installation and configuration process of SAS Decision Services. The location is the database that is indicated in the $User_Log_JDBCConnectionResource. During configuration, it is possible to set a schema for the user log. If no schema is specified, the default schema for the credentials that were set in the $User_Log_JDBCConnectionResource is used.

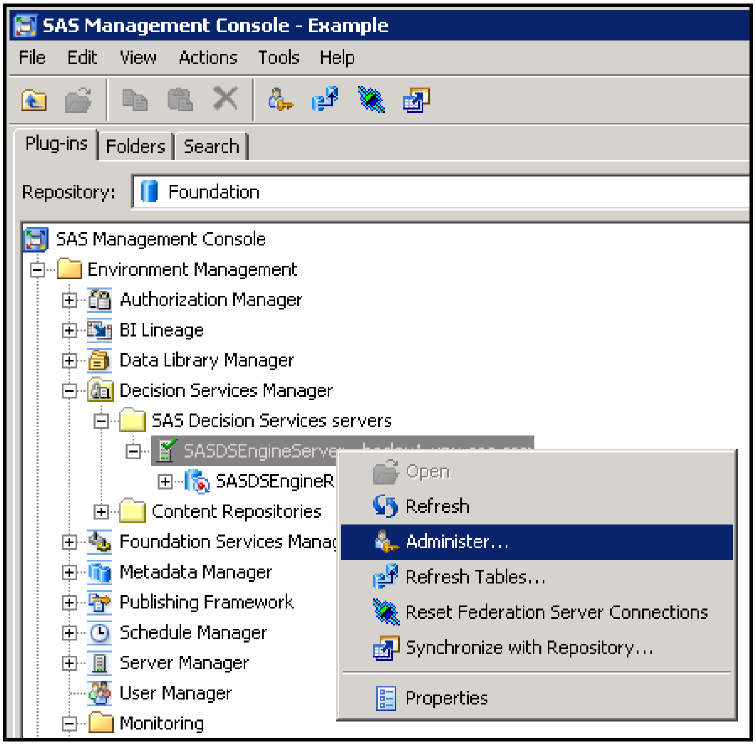
To enable the user log, follow these steps:

1 Launch SAS Management Console.

2 Expand **`Decision Services Manager`** and **`SAS Decision Services servers`**.

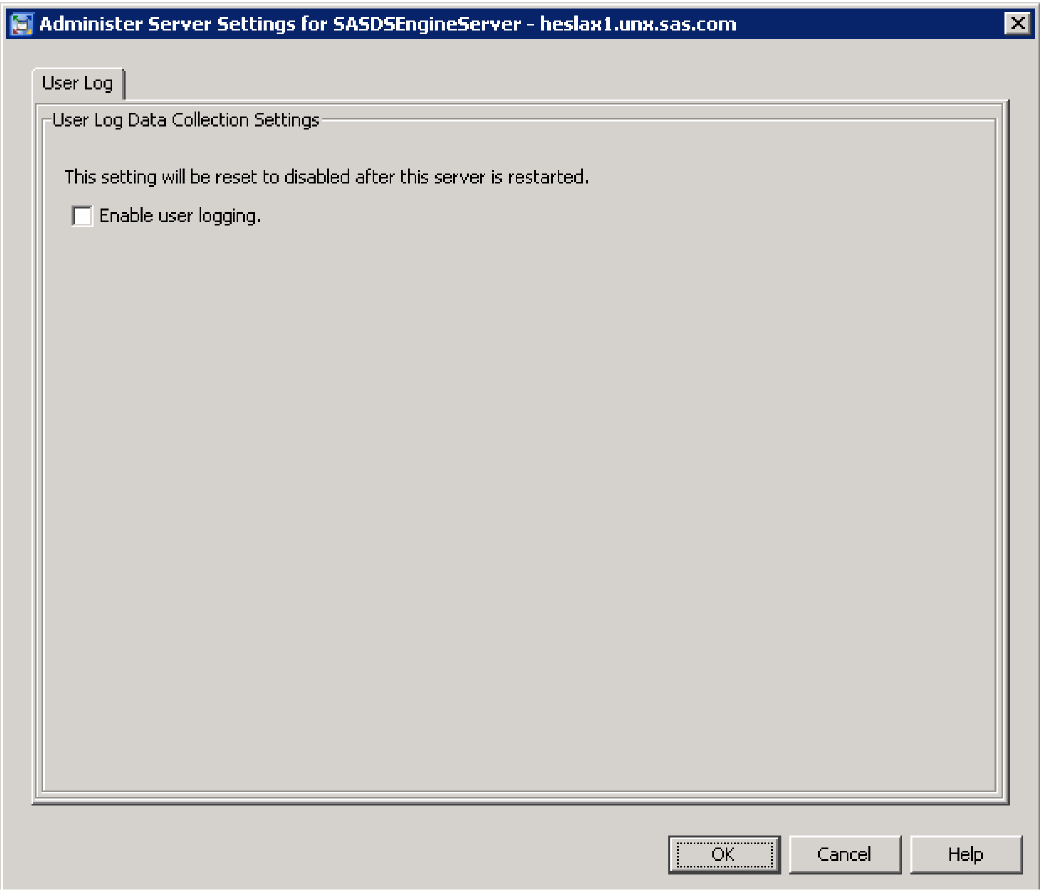


3 Right-click the system that you want to collect performance data for, and select **Administer**.



4 On the **User Log** tab, select the check box for **Enable user log data collection of decision flows executing in the SAS Decision Services server**.

Note: Enabling data collection affects performance. To disable data collection to a log, clear this box.



Administer Server Settings for SASDSEngineServer - heslax1.unx.sas.com

User Log Data Collection Settings

This setting will be reset to disabled after this server is restarted.

□ Enable user logging.

4

Environments and Resources

Repositories

Overview

Before using any of the advanced SAS Decision Services functions, such as creating a new repository, make sure that you understand how to administer content repositories.

Use the advanced functions in the Decision Services Manager plug-in to SAS Management Console to create and delete repositories.

About Repositories

Repositories contain decision flows and their building blocks. These building blocks include events, activities, global variables, and system resources. You specify a repository as a development, testing, or production repository.

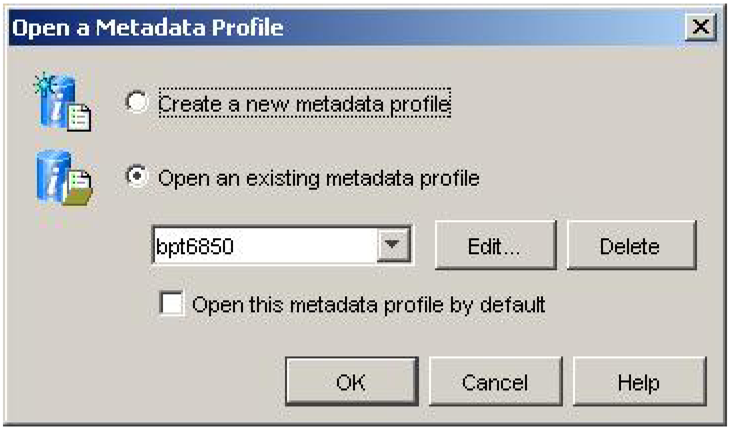
A repository does not have to be associated with a server; it can be used simply as a storage area for artifacts.

A repository resides in SAS Metadata Repository.

Create a Repository

To create a new SAS Decision Services repository, follow these steps:

1 Log in to SAS Management Console. Select the metadata profile that is associated with the SAS Metadata Repository where you want to create your repository. For more information about metadata profiles, see the SAS Management Console Help.



○ Create a new metadata profile

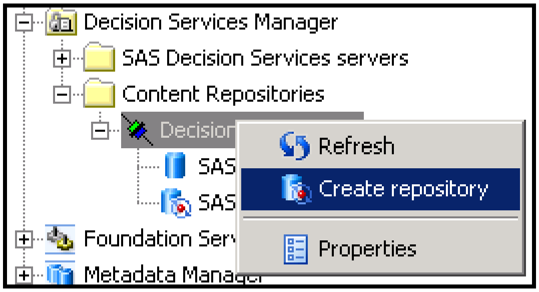
● Open an existing metadata profile

bpt6850

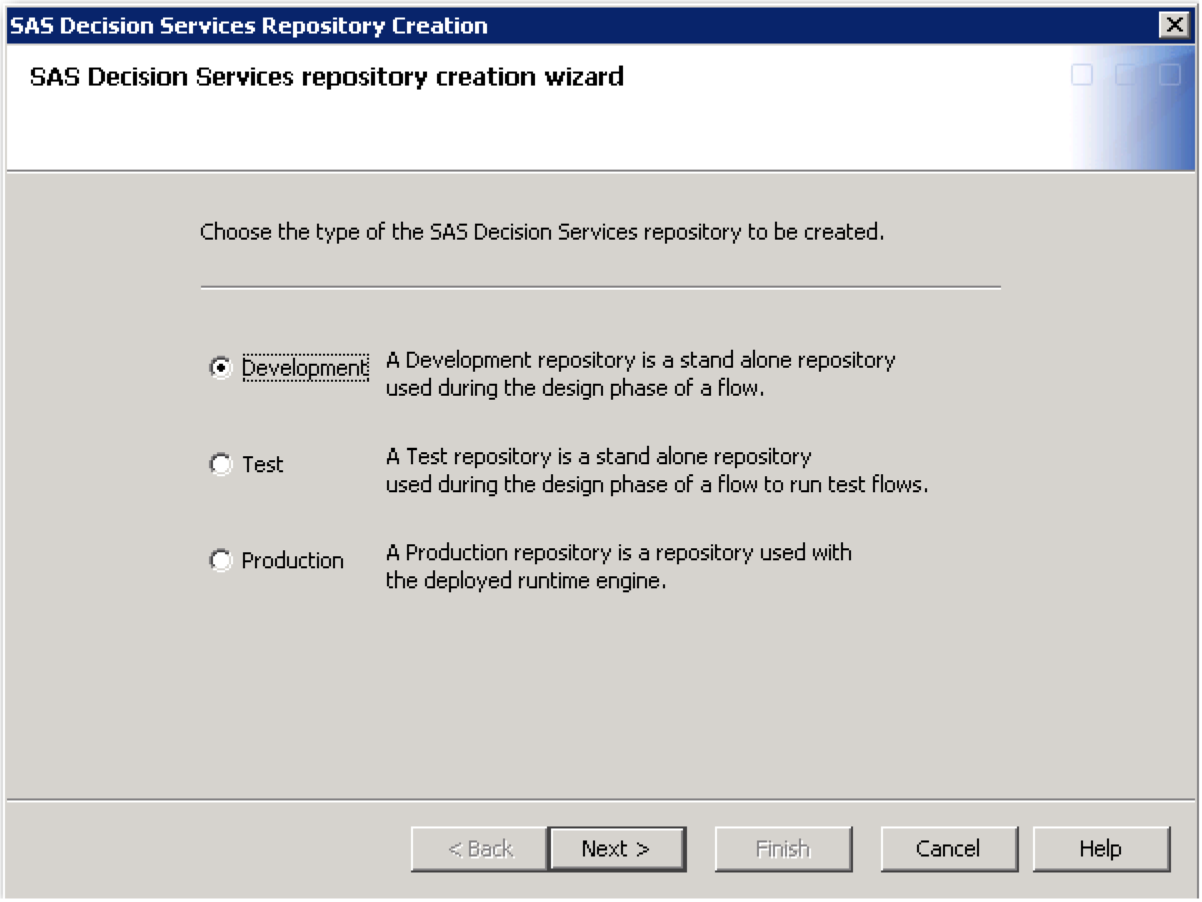
□ Open this metadata profile by default

2 Expand **Decision Services Manager** and **Content Repositories**.

3 Right-click the content mapped folder where you want to create your repository, and select **Create repository**.



4 Choose either a development, test, or production repository. Click **Next**.


SAS Decision Services Repository Creation

SAS Decision Services repository creation wizard

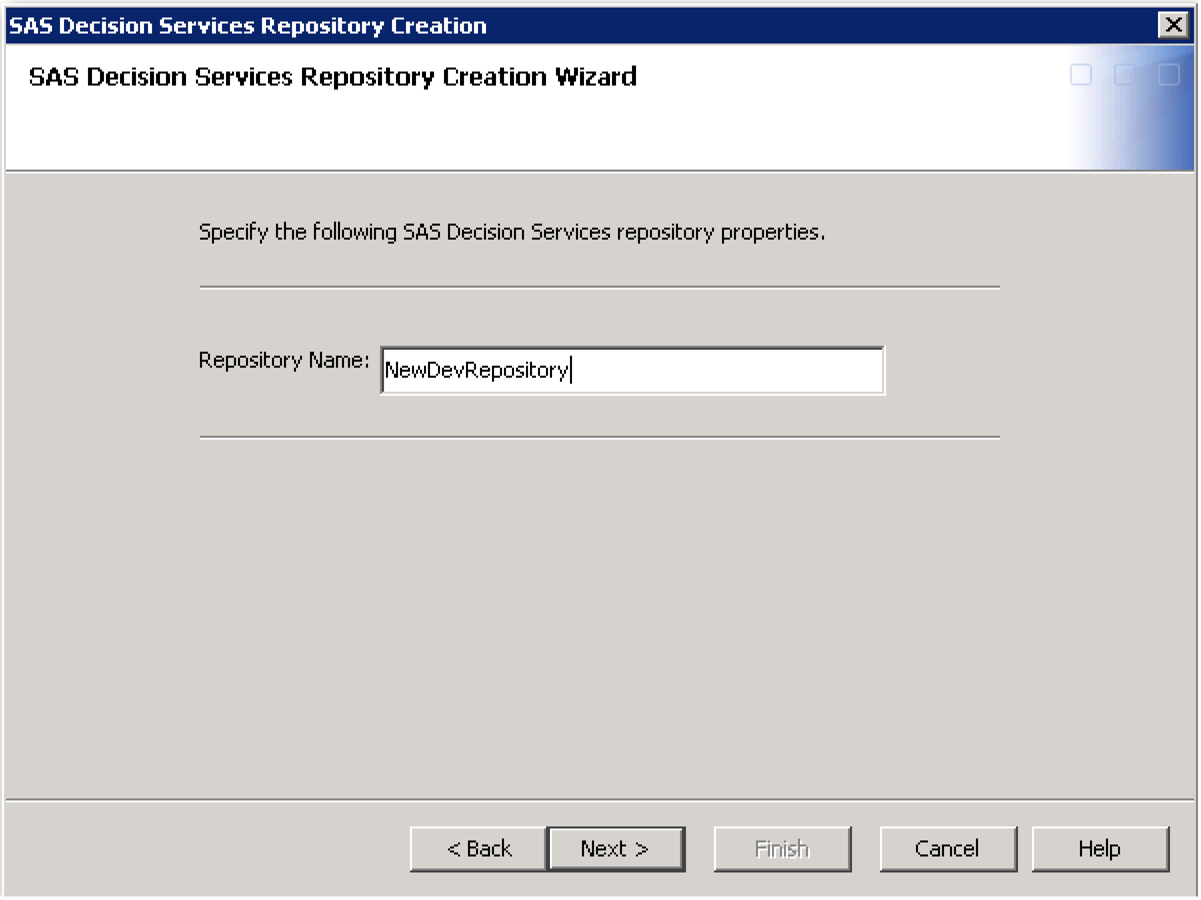
Choose the type of the SAS Decision Services repository to be created.

● Development A Development repository is a stand alone repository used during the design phase of a flow.

○ Test A Test repository is a stand alone repository used during the design phase of a flow to run test flows.

○ Production A Production repository is a repository used with the deployed runtime engine.

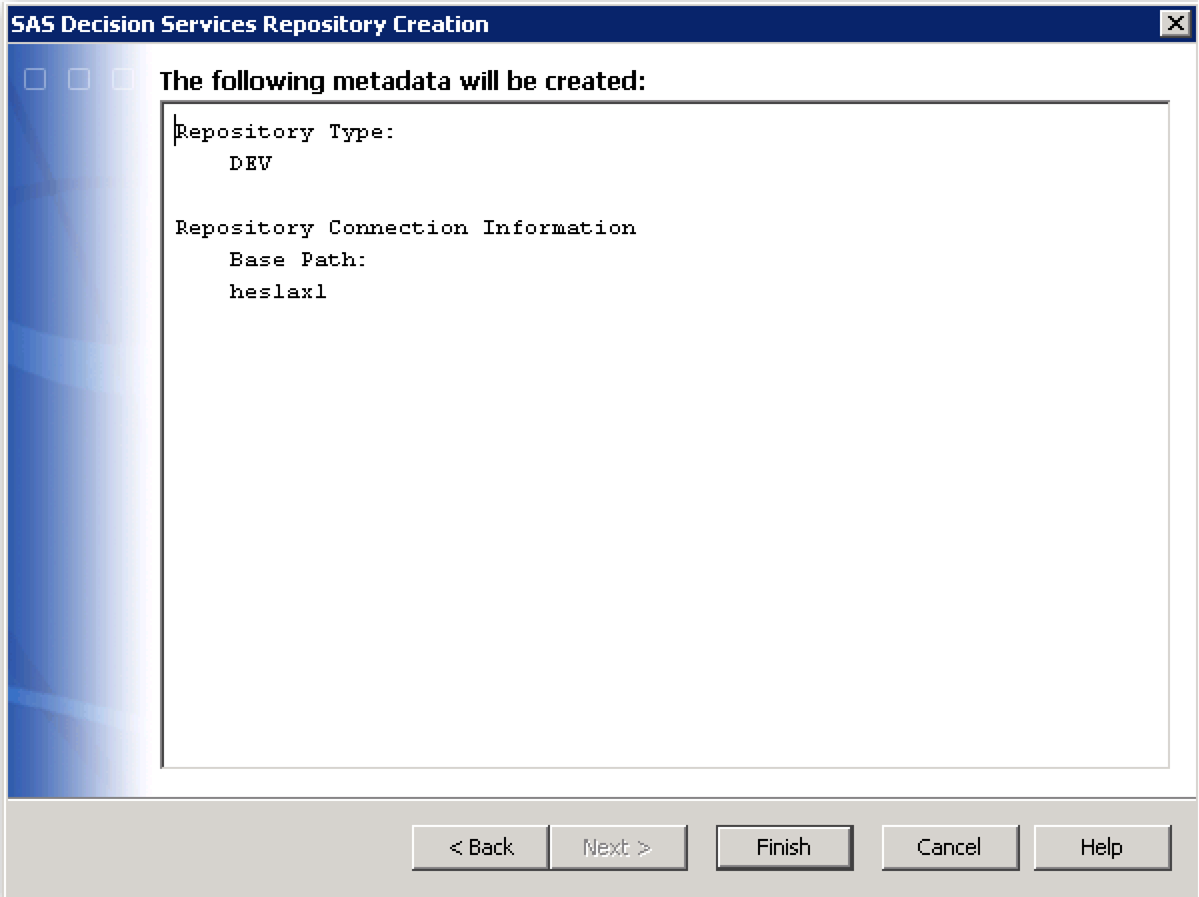
5 Enter a name for your new repository. The following example shows the creation of a new repository called NewDevRepository. Click **Next**.


SAS Decision Services Repository Creation
SAS Decision Services Repository Creation Wizard
< Back
Next >
Finish
Cancel
Help

Specify the following SAS Decision Services repository properties.

Repository Name: NewDevRepository

6 Review the information for accuracy. Click **Finish**.

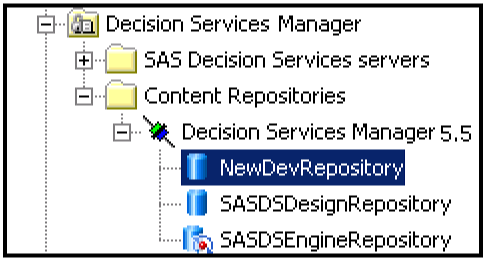


The following metadata will be created:

```
Repository Type:
    DEV

Repository Connection Information
    Base Path:
    heslaxl
```

7 Verify that your repository was created correctly. Expand your repository folder. If your repository was created successfully, you should see the following folders.



Decision Services Manager
SAS Decision Services servers
Content Repositories
Decision Services Manager 5.5
NewDevRepository
SASDSDesignRepository
SASDSEngineRepository

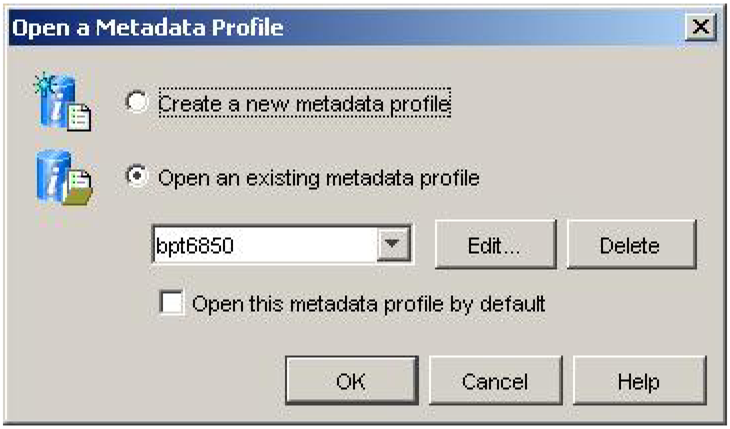
A repository is bound to an engine or design server when that server is installed and configured. For more information, see Chapter 7, “Installation,” on page 103.

Delete a Repository

CAUTION! Deleting a repository is an irreversible operation.

To delete a repository:

1 Log in to SAS Management Console. Choose the metadata profile that is associated with the SAS Metadata Repository that contains the repository to delete.



Open a Metadata Profile

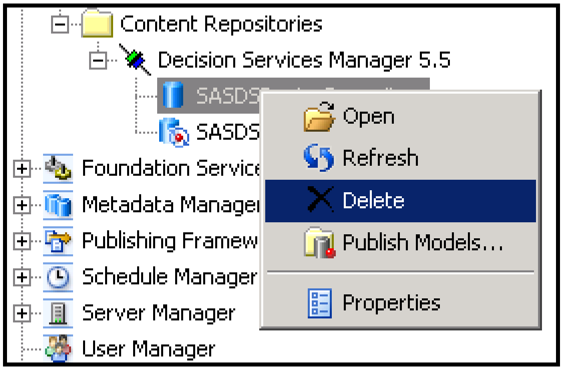
○ Create a new metadata profile

● Open an existing metadata profile

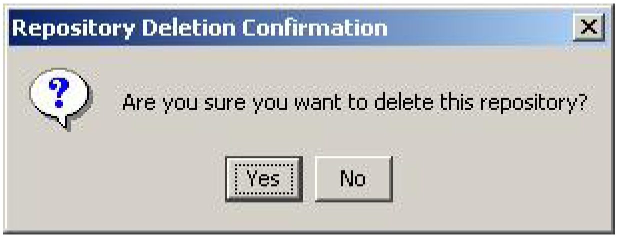
bpt6850

□ Open this metadata profile by default

2 Expand **Decision Services Manager** and **Content Repositories**. Right-click the repository that you want to delete and select **Delete**.



3 Verify your intent to delete the repository by clicking **Yes**.



Repository Deletion Confirmation

Are you sure you want to delete this repository?

System Resources

Overview

System resources enable the engine server to access and interact with resources such as SAS servers or database servers that are outside the J2EE middle tier. Activities reference the system resources by name.

For example, many activities run a SAS DS2 program to produce results. The middle tier portion of these activities must communicate with a DataFlux Federation Server. A system resource type named JDBC Connection provides the information that is needed to facilitate such communications. More specifically, the JDBC Connection system resource contains information that is needed by a SAS activity to execute a DS2 program running on the DataFlux Federation Server.

Also, the JDBC Connection system resource is used to connect to database servers for use in the General I/O activity. These resources point directly to the database using the database’s own JDBC driver.

The Web service system resource is used to connect to external Web services. By providing the end point URL, SAS Decision Services can use the Web service that is pointed to.

Activities use a name to reference system resources instead of containing the resource information directly. Thus, flows are portable between systems. The product supports configurable development, test, and production environments. Typically, the sets of back-end SAS servers that are used by development, test, and production environments are different. System resources enable the correct set of servers to be used in each environment without modification of flows or activities. That is, each environment contains system resources that have the same names. However, the information that is contained by these system resources differs from environment to environment.

About JDBC Connection System Resources

JDBC Connection system resources are used by both SAS activities that execute DS2 programs and by General I/O activities that access database records. The basic fields are listed in step 5 of the following section on page 56. In the case of General I/O, the Connection Options value is not required.

To connect to SAS DATA sets and to execute DS2 SAS activities, a DataFlux Federation Server connection must be used. Therefore, the Connection Options value must be set.

Advanced options are available that allow for the fine tuning of the connection and statement pools used by SAS Decision Services. These values should be set to appropriate values based on the hardware being used. A list of these options appears in "Tuning Controls" on page 134.

To allocate computing resources efficiently, set up more than one DataFlux Federation Server in the server tier. Every server within a given cluster processes the same activity set. The following example illustrates this concept.

Imagine there are two DataFlux Federation Servers: A and B. Suppose also that you have some scoring activities that are very computationally intensive. In order to allocate hardware resources efficiently, scoring activities should be assigned to server A and all other activities to server B. This can be accomplished with two JDBC Connection system resources, one for each DataFlux Federation Server.

The following activity types use the JDBC Connection system resource:

- SAS Activity
- Scoring Activity
- General I/O Activity

Typically one JDBC Connection resource specifies which DataFlux Federation Server each SAS Activity uses, but another JDBC Connection resource specifies which database the General I/O activity has access to.

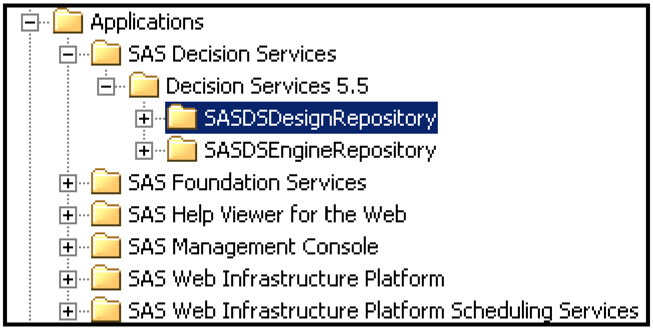
Because decision flows must be portable, DataFlux Federation Server connection information is assigned to activities indirectly by associating them with JDBC Connection system resources. This mechanism allows a flow, without modification, to use an entirely different set of DataFlux Federation Servers in the production environment than it used in the development environment.

Specify a New System Resource as a JDBC Connection

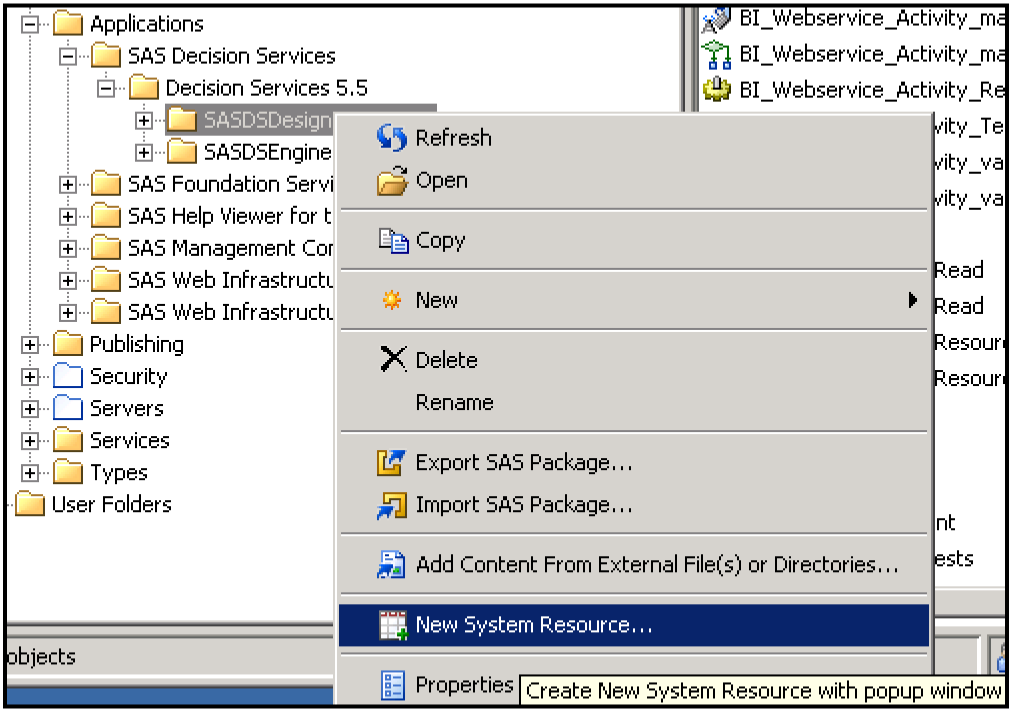
To create a new system resource as a JDBC Connection, click the **Folders** tab, and follow these steps:

1 Expand **System** ▶ **Applications** ▶ **SAS Decision Services** ▶ **Decision Services 5.5**.

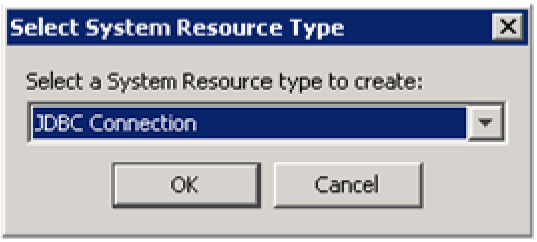
2 Right-click a repository folder such as **SASDSDesignRepository**.



3 Select **New System Resource** from the drop-down menu.

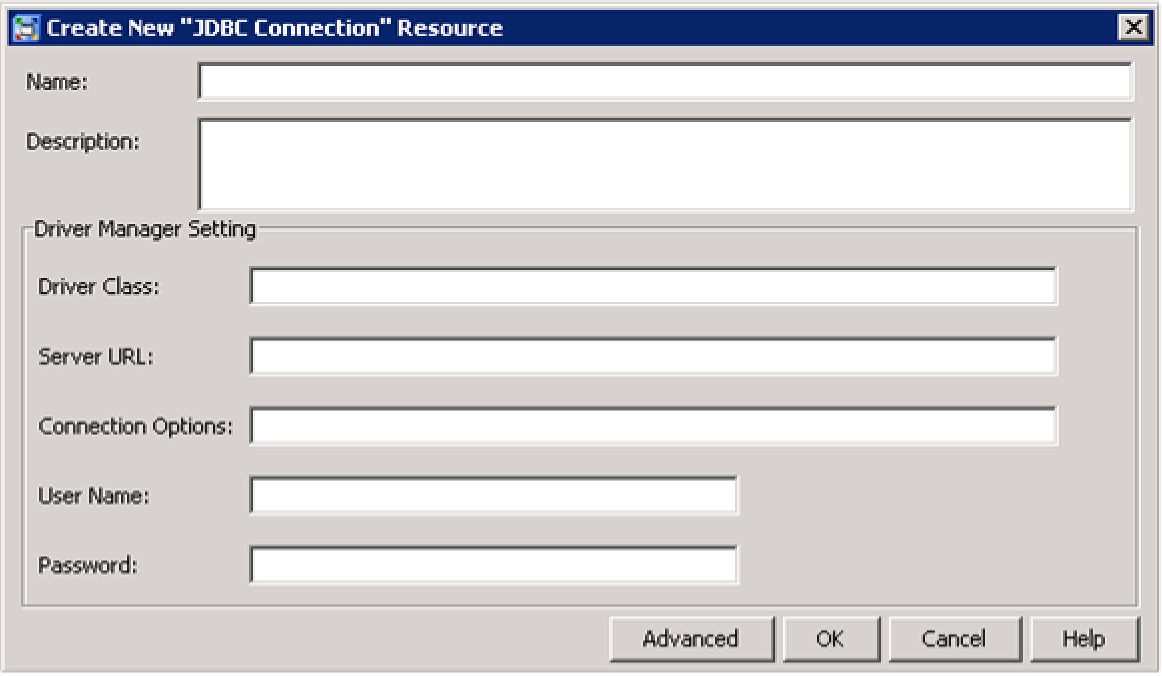


4 Select **JDBC Connection**.



Select a System Resource type to create: JDBC Connection

5 Complete any required fields in the dialog box that appears.



The terms and definitions that follow are also listed in the Help for this dialog box.

Name
: specifies the name of the system resource. It has a 60-character maximum length. Spaces are allowed.

Description
: (optional) might include the SAS activity or server cluster for which you plan to use this SAS connection. Description has a 200-character maximum length.

Driver Class
: specifies the Java class name of the database or DataFlux Federation Server driver. To create a resource for accessing database tables, use the class name of the driver that is provided by your database vendor. If you are unsure of what driver class name to use, see your system administrator.

Table 4.1 *Examples for the Driver Class Field*

Database	**Class Name**
Oracle	oracle.jdbc.driver.*OracleDriver*

SQL Server	com.microsoft.sqlserver.jdbc.*SQLServerDriver*
Teradata	com.teradata.jdbc.*TeraDriver*
DB2	com.ibm.db2.jcc.*DB2Driver*

Note: To create a system resource for accessing a DataFlux Federation Server, enter `com.sas.tkts.TKTSDriver`.

Server URL

is a database URL of the form jdbc:subprotocol:subname. See your system administrator for the URL that references your database installation. To create a system resource for executing DS2 activities, use the URL form `jdbc:sastkts://host:port`, where *host* and *port* reference your DataFlux Federation Server installation.

Table 4.2 Examples for the Server URL Field

Database	URL
Oracle	jdbc:oracle:thin:@//[*OraclePath*]
SQL Server	jdbc:sqlserver://[*SQL Server Host*]
Teradata	jdbc:teradata://[*Teradata Server Host*]
DB2	jdbc:db2:[*DB2 Server*]

Connection Options

(optional) use this field to create a resource for executing DS2 activities. The connection options should be in the form of `DRIVER=TSSQL;CONOPTS=(DSN=Federation Server DSN)`.

For direct-to-database connections (general I/O), see the documentation for the specific database, to determine what options are available. With direct-to-database connections, the connection options are optional.

User Name
: (optional) is used to connect to the database or DataFlux Federation Server that is specified in Server URL.

Password
: (optional) is the password that is used to connect to the database or to the DataFlux Federation Server that is specified in Server URL, along with the user name.

(optional) Click **Advanced** to access connection and statement pool tuning controls. See the Performance Tuning chapter on page 134 for more information.

Specify a New System Resource as a Web Service Connection

The Web Service Activity type does not use the SAS server tier for processing. Instead, it makes a direct request to the Web service as specified by the Web Service Connection system resource.

To specify a Web Service Connection as a system resource, follow steps 1–3 in “Specify a New System Resource as a JDBC Connection” on page 55, and continue with these steps:

1 Select **Web Service**.

2 Complete any required fields in the dialog box that appears.

The terms and definitions that follow are also listed in the Help for this dialog box.

Name
: specifies the name of the system resource. Name has a 60-character maximum length; spaces are allowed.

Description
: (optional) might specify the Web Service activity that you plan to use this system resource for. Description has a 200-character maximum length.

WSDL URL
: (required) specifies the URL of the target Web service. If the WSDL URL begins with `https`, then the **User Name** and **Password** fields are also required.

Note: You must enter a valid URL for the WSDL. If the URL contains spaces and other disallowed characters, they must be encoded.

Host

(optional) specifies the proxy server that forwards client requests to other servers. See your system administrator for whether your installation uses a proxy server, and if so, what host name you should use.

Port

(optional) specifies the port that is used by the proxy server.

User Name

If the WSDL URL begins with `https` (indicating that security is enabled), then this field specifies your user name.

Password

If the WSDL URL begins with **`https`** (indicating that security is enabled), this field specifies your user password.

After you click **OK**, the new Web Service Connection system resource should appear in the repository.

Library Resources

Overview

Library resources provide two distinct capabilities:

- To define alias names for database schemas
- To specify tables to cache in read-only memory

Note: Both of these features are optional and can be used together or separately.

(Optional) Define a Schema Alias

SAS Decision Services supports the optional use of aliases to reference database schemas.

For example, suppose your database has a schema called DDA, for direct-deposit accounts, and the SAS programs in your organization reference this schema by using a libref called ACCOUNTS. Because SAS Decision Services accesses data from your database directly, without going through SAS/ACCESS, internally the SAS Decision Services engine must use the actual schema name to access the tables within the schema.

For consistency with SAS, or to define user-friendly names, you might want to create an alias for DDA called ACCOUNTS by using a library resource.

Your SAS Decision Services repository can contain zero or more library resources. You must create a library resource for each schema alias that you want to define.

To specify a library resource, follow steps 1–3 in "Specify a New System Resource as a JDBC Connection" on page 55, and continue with these steps:

1 Select **Library**.

2 Complete any required fields in the dialog box that appears.

   The terms and definitions that follow are also listed in the Help for this dialog box.

   Name
      specifies the name of the library resource and the alias name to use. Host has a 60-character maximum length. Spaces are allowed.

   Description
      (optional) might describe the schema referenced by this library resource. Description has a 200-character maximum length.

   Schema Name
      the actual schema name defined to the database. Description has a 200-character maximum length.

Connection Resource
: select the JDBC Connection system resource that you created above from the drop-down list.

(Optional) Specify Tables to Cache in Memory

SAS Decision Services provides a memory cache for hosting read-only tabular data. This is an optional, performance-enhancing feature. Data in memory can be accessed much faster than data on disk. Good candidates for caching are tables that change very infrequently, but are referenced frequently. For example, a table of automobile part numbers, names, and descriptions would be a good candidate.

To specify tables to be cached, create a library resource for the schema that contains the table, and fill in the fields under **Cached Tables**. Add a row for each table to be cached in memory. You must create a library resource for each schema that contains tables to cache.

Note: Sufficient memory must exist on each middle-tier engine server to hold all tables specified for caching. Otherwise, run-time errors result.

Table Name
: name of table to be cached in memory.

Columns Clause
: a comma-separated list of columns to cache, or * for all columns.

Order By Clause
: (optional) a comma-separated list of columns to sort the in-memory table by.

Cached
: if checked, the table is cached. Otherwise, the table is not cached.

If you do not want to create an alias for the name of the schema that contains tables to cache, enter the same value in both the `Name` and `Schema Name` fields.

5

Decision Services Activities

Overview

SAS Decision Services provides a rich set of activities for constructing decision flows that automate real-time decisions and actions. Activities perform work actions, such as executing SAS programs on a SAS server, storing and accessing information from a relational database, sending Web service requests to external systems, and executing scoring models.

If your organization has a special processing need that is not covered by the provided activity set, new activities can be added. This is accomplished by developing custom SAS code and publishing it to the SAS Decision Services environment. The activity publishing step assembles metadata. Metadata is necessary for the activity to be recognized by a SAS Decision Services engine and to be rendered in a client environment, such as SAS Customer Intelligence Studio.

SAS Decision Services uses the following classifications of configurable activities:

- SAS activity
- Web service activity
- general I/O activity
- scoring activity
- middle-tier code activities

The activity type that can be used to extend SAS Decision Services functionality is known as the SAS activity. A SAS activity consists of a SAS program and an activity XML file that describes the methods that are supported by that activity and the system resources that are used by that activity.

SAS DATA step programming skills are required to develop SAS code that runs as an activity. For assistance with custom activity development or publishing, contact your on-site SAS support personnel.

SAS Activities

What Is a SAS Activity?

SAS activities are powerful tools for expanding the functionality of SAS Decision Services. A SAS activity corresponds to a DS2 package. A DS2 package is a set of associated methods that perform specific functions.

Creating a New SAS Activity

Overview

To create a new SAS activity, first create a DS2 package that contains the SAS code to be executed. After the package has been completed and stored in the database, create the activity metadata that corresponds to it. After that, create flows that include the SAS activity.

Efficiency Considerations

Consider efficiency first when developing SAS activity code. If your decision flows are required to provide an immediate response, avoid implementing long-running processes such as joins, non-indexed searches, and other expensive database queries. Remember that a decision flow executes no faster than the cumulative speeds of the activities that it contains.

Create a DS2 Package

Create your DS2 package in an interactive SAS session. This method enables you to conduct immediate testing to be sure the code is correct. DS2 packages are created using PROC DS2. For more information about PROC DS2, see *SAS 9.3 DS2 Language Reference* and *SAS 9.3 DS2 Language Reference: Getting Started*, available at `http://support.sas.com/documentation/solutions/ds2/`.

1 Set the CONN option in your PROC DS2 statement to point to your database:

```
proc ds2 nolibs conn="driver=remts;server=your_Fed_Server;port=21032;
protocol=bridge;uid=user;pwd=password;
conopts=(DSN=Fed_Server_DSN)";
 package my_pkg /overwrite=yes sas_encrypt=yes;
   method echo_string(varchar(32767) in_string, in_out varchar out_string);
     out_string=in_string;
   end;
 endpackage;
 run;
quit;
```

This code creates a package that is called my_pkg that contains one method, echo_string, and stores it in the database that is pointed to by the DataFlux Federation Server DSN. SAS activity methods must be coded as void functions in DS2. Output parameters must be marked with the in_out tag, which causes their values to be returned to the middle tier after method execution.

You can test your package in your interactive SAS session by using a DS2 TABLE_NULL statement:

```
proc ds2 nolibs conn="driver=remts;server=your_Fed_Server;port=21032;
protocol=bridge;uid=user;pwd=password;
conopts=(DSN=Fed_Server_DSN)";
 table _null_;
   method init();
     dcl package my_pkg echo();
     dcl varchar(32767) out_string;
      echo.echo_string('String to echo', out_string);
     put out_string=;
   end;
 endtable;
run;
quit;
```

2. After changing and republishing an existing DS2 activity, you must notify any running engines that the activity has changed. Existing DataFlux Federation Server connections in a running engine continue to use the original activity until you have reset the DataFlux Federation Server connections or restarted the engine. To reset the connections, go to the Decision Services Manager plug-in in SAS Management Console, and perform the following steps:

   a. On the **Plug-ins** tab, select **Environment Management** ▶ **Decision Services Manager** ▶ **SAS servers**.

b Right-click the engine server object, and select **Reset Federation Server Connections**.

c Confirm that you want to reset the DataFlux Federation Server connections.

Create SAS Activity XML

SAS Decision Services client applications, such as SAS Customer Intelligence, provide an interface for entering activity metadata. For more information, see your client application's documentation.

1 Using the client application, create a new activity.

2 Give the activity a name to match the DS2 package name that was created earlier.

3 Enter a description that includes you as the owner and that describes the purpose of the activity. This is good practice that enables you to better manage your files.

4 Enter methods that match each method in your DS2 package. The order of the parameters in the method is important and must match the order of parameters in the DS2 package method.

5 Select the system resource that contains your DataFlux Federation Server connection information.

Data Type Mappings

The following table lists the SAS Decision Services data types and maps them to specific DS2 data types.

Table 5.1 *Data Types*

SAS Decision Services Data Type	DS2 Data Type
String	Varchar
Int	Bigint
Float	Double
Boolean	Integer

DateTime	Double

Out of the Box SAS Activity DS2 Package

Overview

Several DS2 packages are provided out of the box, in the following location ***<Lev Config Dir>*\Applications \SASDecisionServicesServerConfig5.5\SASCode**. Some of those SAS files are utilities to help you build your own SAS activities, and some are sample SAS activities.

Utility Packages

tap_hash

This package is a simple extension of the DS2 hash object. Each new package does not need to declare its own hash extension package; this one is provided for everyone’s use. Reference this package by specifying the database catalog name, a period, and then `tap_hash`.

tap_logger

This package provides a basic logging capability. By using this package, the DS2 package author can write messages to the SAS_Activity.log file that the DataFlux Federation Server points to. Whether a message is printed to the log depends on the logging level that is set in the log configuration file.

Here are the available logging methods:

- trace(varchar msg); - Log a trace level message.
- debug(varchar msg); - Log a debug level message.
- info(varchar msg); - Log an information level message.
- warn(varchar msg); - Log a warning level message.
- err(varchar msg); - Log an error level message.
- fatal(varchar msg); - Log a fatal level message.

tap_array

SAS Decision Services array objects are passed to a DS2 method as an encoded string (varchar) parameter. Use the tap_array package to decode the string. Empty array objects can also be created and populated by your custom SAS activity code. This package provides an encode() method that can be called to create an encoded string version of the current array. This is the array that is to be returned to the SAS Decision Services engine.

Here are the available methods:

- tap_array(); - Constructs an empty array.

  Note: set_type must be called immediately after using this constructor.

- tap_array(varchar input_array); - Constructs an array that is initialized using the encoded input_array string.
- set_type(varchar type); - Sets the type of array. Choose one of the following types: STRING, INT, FLOAT, BOOLEAN, or DATETIME. This method is not case sensitive.
- type() returns varchar; - Returns the type of array. Choose one of the following: STRING, INT, FLOAT, BOOLEAN, or DATETIME.
- encode() returns varchar; - Encodes this array into a string for return to the SAS Decision Services engine.
- add(varchar element); - Appends the specified element to the end of this array.
- add(int element); - Appends the specified element to the end of this array.
- add(double element); - Appends the specified element to the end of this array.
- add(int index, varchar element); - Inserts the specified element at the specified position in this array.
- add(int index, int element); - Inserts the specified element at the specified position in this array.
- add(int index, double element); - Inserts the specified element at the specified position in this array.

- addAll(package tap_array in_array); - Appends all of the elements in the specified array to the end of this array.
- clear(); - Removes all of the elements from this array.
- set_null(); - Sets this array to null.
- getString(int index) returns varchar; - Returns the element at the specified position in this array.
- getInt(int index) returns int; - Returns the element at the specified position in this array.
- getDateTime(int index) returns double; - Returns the element at the specified position in this array.
- getBoolean(int index) returns int; - Returns the element at the specified position in this array.
- getFloat(int index) returns double; - Returns the element at the specified position in this array.
- isEmpty() returns int; - Returns 1 (true) if this array contains no elements, 0 (false) otherwise.
- delete(int index); - Deletes the element at the specified position in this array.
- setString(int index, varchar element); - Replaces the element at the specified position in this array with the specified element.
- setInt(int index, int element); - Replaces the element at the specified position in this array with the specified element.
- setFloat(int index, double element); - Replaces the element at the specified position in this array with the specified element.
- setDateTime(int index, double element); - Replaces the element at the specified position in this array with the specified element.
- setBoolean(int index, int element); - Replaces the element at the specified position in this array with the specified element.
- size() returns int; - Returns the number of elements in this array.

tap_table

SAS Decision Services table objects are passed to a DS2 method as an encoded string (varchar) parameter. To decode the string, you must use the tap_table package. Empty table objects can also be created and populated in your custom SAS activity code. This package provides an encode() method that can be called to create an encoded string version of the current table that is returned to the SAS Decision Services engine.

Here are the available methods:

- tap_table(); - Creates an empty table.
- tap_table(varchar input_table); - Creates a table that is initialized with the input table string.
- encode() returns varchar; - Encodes the table into a string that can be passed back to the SAS Decision Services engine.
- add_column(varchar name, varchar type); - Adds a column of the given type to the table.
- add_row(); - Adds a new row to the table, all values are set to null.
- add_row(int rows); - Adds the specified number of rows to the table, all values are set to null.
- column_count() returns int; - Returns the number of columns in the table.
- row_count() returns int; - Returns the number of rows in the table.
- column_name(int index) returns varchar; - Returns the name of the column at the given ordinal.
- column_type(int index) returns varchar; - Returns the type of the column at the given ordinal.
- column_type(varchar name) returns varchar; - Returns the type for the given column.
- delete_column(varchar name); - Removes the given column from the table.
- delete_row(int row); - Removes the given row from the table.

- getString(varchar col_name, int row) returns varchar; - Retrieves the string value from the given column at the given row.
- getInt(varchar col_name, int row) returns int; - Retrieves the int value from the given column at the given row.
- getBoolean(varchar col_name, int row) returns int; - Retrieves the Boolean value from the given column at the given row.
- getFloat(varchar col_name, int row) returns double; - Retrieves the float value from the given column at the given row.
- getDateTime(varchar col_name, int row) returns double; - Retrieves the datetime value from the given column at the given row.
- setString(varchar col_name, int row, varchar element); - Sets the string value from the given column at the given row.
- setInt(varchar col_name, int row, int element); - Sets the int value from the given column at the given row.
- setFloat(varchar col_name, int row, double element); - Sets the float value from the given column at the given row.
- setDateTime(varchar col_name, int row, double element); - Sets the datetime value from the given column at the given row.
- setBoolean(varchar col_name, int row, int element); - Sets the Boolean value from the given column at the given row.
- set_null(); - Sets the table to null.

Sample Package

sas_activity_tests

This is a sample package that can be used for validation and testing. Here are the available methods:

echo_string

Signature - (varchar(32767) in_string, in_out varchar out_string)

Description - Echoes the input string to the output string.

echo_int

Signature - (int in_int, in_out int out_int)

Description - Echoes the input int to the output int.

echo_float

Signature -(double in_float, in_out double out_float)

Description - Echoes the input float to the output float.

echo_boolean

Signature - (int in_boolean, in_out int out_boolean)

Description - Echoes the input Boolean to the output Boolean.

echo_datetime

Signature - (double in_datetime, in_out double out_datetime)

Description - Echoes the input datetime to the output datetime.

echo_scalars

Signature - (varchar(32767) in_string, int in_int, double in_float, int in_boolean, double in_datetime, in_out varchar out_string, in_out int out_int, in_out double out_float, in_out int out_boolean, in_out double out_datetime)

Description - Echoes the input values to the output values.

echo_array

Signature - (varchar(32767) in_array, in_out varchar out_array)

Description - Echoes the input array to the output array.

echo_table

Signature - (varchar(32767) in_table, in_out varchar out_table)

Description - Echoes the input table to the output table.

variable_test

Signature - (varchar(32767) in_string, int in_int, double in_float, int in_boolean, double in_datetime, varchar(32767) in_array, varchar(32767) in_table, in_out varchar out_string, in_out int out_int, in_out double out_float, in_out int out_boolean, in_out double out_datetime, in_out varchar out_array, in_out varchar out_table)

Description - Edits each of the input values and sets them in the output values.

- out_string - The result of reversing in_string. For example, “abc” becomes “cba.”
- out_int - The result of in_int + 2.
- out_float - The result of in_float + 1.11.
- out_boolean - The negation of in_boolean - true = false and false = true.
- out_datetime - The result of out_datetime + 1 day.
- out_array - The reverse array order of in_array - String1, String2, String3 becomes String3, String2, String1.
- out_table - The input table with the row order reversed, 100 added to each column of type int, 222.222 added to each column of type float, 6 days added to each column of type datetime, the string reverse for each column of type string, and the negation for each column of type Boolean.

Reading Database Tables from a Custom SAS Activity

The preferred vehicle for accessing a database is the General I/O activity. However, there might be times when it is advantageous for custom SAS code to do so.

Note: SAS activities can read only database records. They cannot insert or update records.

To enable SAS activity code to read from a database, you must first create a federated DSN. Federated DSNs contain a list of standard DSNs, enabling access to more than one data source. By referencing the federated DSN in your connection string, you gain access to all of the catalogs and schemas that are referenced by the contained DSNs.

Because DS2 packages are stored in SAS data sets, your federated DSN must include BASE_DSN as well as any additional DSNs that reference the database catalogs, schemas, and tables you want to access.

To create a federated DSN, connect to Federation Server Manager and log on to your federation server definition with a user ID that has administrative privileges, and follow these steps:

1 With the Federation Server definition selected, click the **Data Source Names** tab.

2 From the drop-down list, select **New Federated Data Source Name**.

3 Enter the name and description for the federated DSN, and click **Next**.

4 From the drop-down list, select **Add Data Source Names**.

5 Select the DSNs that you want to connect to with this federated DSN, and click **OK**.

6 When you return to the Members screen, click **Next**.

7 It is recommended that you keep the default security setting, and click **Next**.

8 When you have reviewed the information about the Summary screen, click **Finish**.

You can test your federated DSN by modifying the following SAS program:

```
proc ds2 Conn="driver=remts;server=your_server;port=your_port;protocol=bridge;
  uid=admin_userid;pwd=admin_password;conopts=(DSN=base_ora_dsn)";table _null_;

  method run();
      set DATABASE_CATALOG.SCHEMA.TABLE;
      put a_column= another_column=;
  end;

endtable;

run;
quit;
```

Custom SAS activities are implemented as DS2 packages. To enable reading from within a package method, the “dataset()” method was added to the DS2 hash object. This method takes an SQL SELECT statement as an argument and populates the hash object with the corresponding result set.

```
method compute();
    dcl package hash h();
    dcl package hiter hi(h);
```

```
      dcl int rc;

      h.definekey('clientid');
      h.definedata('hhid');
      h.definedata('income');
      h.dataset(
  '{select clientid, hhid, income from DSORA.MAFUNC.CUSTOMER1;}');
      h.definedone();

      rc = hi.first();
      do while(rc = 0);
        …do something with the data…
        rc = hi.next();
      end;

    end;
```

For more information, see the *SAS 9.3 DS2 Language Reference*.

Web Service Activities

Invoking External Web Service Activities

SAS Decision Services functionality can be extended by adding new Web service activities. A Web service activity can invoke an external Web service that requests information to be used downstream in the decision flow. For example, suppose an organization has an inventory system with a Web service interface. It is possible to create a Web service activity that sends a request to the inventory system to check that there is sufficient quantity of a product to extend an offer.

The Web service activity maps leaf-level elements of the XML, for the request and response payloads, to SAS Decision Services process variables of the following data types:

- BOOLEAN
- INT
- FLOAT
- DATETIME

- STRING
- ARRAY OF BOOLEAN
- ARRAY OF INT
- ARRAY OF FLOAT
- ARRAY OF DATETIME
- ARRAY OF STRING

Web service activity supports only transport-level security using SSL (HTTPS).

The Web service activity uses a Web Service Connection system resource. This resource contains the URL of the Web service to invoke. When you publish a new Web service activity, you bind it to a particular Web Service Connection system resource. Create your Web Service Connection system resource before publishing your new Web service activity. For more information about the Web Service Connection system resource, see “Specify a New System Resource as a Web Service Connection” on page 59.

Invoking SAS BI Web Services

SAS BI Web services executes as SAS stored processes in the SAS server tier. SAS BI Web service activity supports an extended set of data types. The standard Web service activity supports the types that are listed above. The BI Web service activity supports the following input and output parameter types:

Table 5.2 *Input Parameter Types*

Stored Process Type	**SAS Decision Services Type**
Numeric (integer)	Int
Numeric (double)	Float
Text	String
Numeric (integer) with name ending in _b	Boolean
Numeric (integer) with name ending in _d	DateTime

Text with name ending in _a	Array
Text with name ending in _t	Table

Table 5.3 Output Parameter Types

Stored Process Type	SAS Decision Services Type
Integer	Int
Double	Float
String	String
Integer with name ending in _b	Boolean
Integer with name ending in _d	DateTime
String with name ending in _a	Array
String with name ending in _t	Table

Tables and arrays are passed in and out of the stored process as encoded strings. An autocall macro, called scencode, is provided to encode these objects.

BI Web service activity supports only transport-level security using SSL (HTTPS).

General I/O Activities

Overview

SAS Decision Services is shipped with a General I/O activity that can read or write to any available database table or SAS data set. A General I/O activity uses a JDBC Connection resource. This resource specifies which database the activity uses. At least one JDBC Connection resource was configured when your system was installed.

Note: It is recommended that you use SAS data sets for reading only.

Operations

Read

Method name: activityMethodName to SCReadTable.

Input Parameters

- G_IO_libraryName - Library or schema name.
- G_IO_tableName - Database table name.
- G_IO_WHERE_Clause - WHERE clause.

Input and Output Parameters

G_IO_Result_Table Result - SAS Decision Services table. On input, this table contains column definitions (name and type). The specified columns are selected from the database, and coerced to the specified type if possible. On output, this table contains the original column definitions plus rows of data that are selected from the database.

Insert

Method name: activityMethodName to SCInsertIntoTable.

Input Parameters

- G_IO_libraryName - Library or schema name.
- G_IO_tableName - Database table name.
- G_IO_Insert_Values - A SAS Decision Services table that contains multiple rows. Corresponding rows are inserted in the database table. Columns that occur in the database but not in this table are set to null or missing.

Input and Output Parameters

None.

Update

Method name: activityMethodName to SCUpdateTable.

Input Parameters

- G_IO_libraryName - Library or schema name.

  If this parameter is blank, the default database schema is used. The JDBC Connection resource that is specified in the General I/O activity definition is used.

  Otherwise, if a JDBC library resource that has the given name is found, that resource is used to get the database schema name and JDBC Connection resource name. If the schema name in the resource is blank, the default database schema is used.

  If a JDBC library resource with a given name is not found, the name is interpreted directly as a database schema name. The JDBC Connection resource that is specified in the General I/O activity definition is used.

  Before SAS Decision Manager 5.5, this parameter specified a SAS libref. This name did not correspond to an actual database schema name. If your installation is earlier than 5.5, it can retain this name, but must add a JDBC library resource that has the same name. That resource can specify the database schema name.

- G_IO_tableName - Database table name.

  A table name in the database schema (default or specific) that is specified by this G_IO_libraryName.

- G_IO_WHERE_Clause - WHERE clause.

  A SAS Decision Services (not SQL) Boolean expression. Logical (AND, OR, NOT), relational (EQ, NE, GT, GE, LT, LE), and arithmetic (+, -, /, *) operators can be used. Here is an example: CustomerInfo.Income GT 50000.0. As in a DATA step, a . (period) denotes null or missing.

  Process parameters can be referenced as: :{*Process parameter name*}. Here is an example: CustomerInfo.LastName EQ :PV_CustomerLastName

  Note: '=' and '!=' are not supported in General I/O WHERE clauses. EQ and NE are used instead.

- G_IO_Update_Values - A SAS Decision Services table that contains one row. The table contains column definitions along with their corresponding values.

Output Parameters

G_IO_Rows_Updated - The number of database rows that are updated.

Library Resources

Using a library resource with the General I/O activity provides a level of indirection to the physical database schema name. It also provides a single location to specify the JDBC Connection resource name for a given schema.

If table caching is desired, a library resource must be used to specify cached tables.

The JDBC Connection resource provides database connection information. The resource name can be specified in a JDBC library resource. If a library resource is not used, the connection resource name is retrieved from the resource that is specified in the General I/O activity definition.

Scoring Activities

Unlike the SAS activity and the Web service activity, the only motivation for adding and configuring new scoring activities is to tune hardware resource use in a DataFlux Federation Server deployment. This is accomplished by directing the processing of the new activities to a different DataFlux Federation Server. For more information about DataFlux Federation Servers, see “About JDBC Connection System Resources” on page 54. A scoring activity uses a JDBC Connection resource. This resource specifies which DataFlux Federation Server the activity uses. For more information, see “System Resources” on page 53. At least one JDBC Connection scoring resource was configured when your system was installed.

Middle-Tier Code Activities

Overview

Code activities execute entirely within the SAS Decision Services engine on the middle tier. As a result, they do not incur the overhead of an out-of-process method call. Therefore, code activities are your best practice for implementing performance-critical business logic. If your client application does not provide a user interface for creating a code activity, you can manually create it.

Here is an example of a code activity in XML format:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<ActivityDefinition
      javaClassName="com.sas.analytics.ph.rt.act.code.CodeActivity"
timeout="0"
      displayName="Code Activity Test"
name="CodeActivityVariableTest"
      xmlns="http://www.sas.com/xml/analytics/rdm-1.1">
     <Description/>
      <Method displayName="sc_variable_test"
name="sc_variable_test">
    <Description>This Method Tests All the different
                 Variable types</Description>
           <Body>
                  <Expression>
                        MyInt = MyInt + 2;
                        MyFloat = MyFloat + 1.11;
                        MyBoolean = NOT MyBoolean;
                        MyString = Reverse(MyString);
                        MyDate = intnx('DTSECOND',MyDate,24*60*60);
                        arraySize = DIM(MyStringArray);
                        do index=1 to (arraySize / 2);
                              temp = MyStringArray[index];
                              MyStringArray[index] =
MyStringArray[arraySize - index + 1];
                              MyStringArray[arraySize - index + 1] =
temp;
                        end;
                  </Expression>
```

```
            </Body>
            <InputParameter array="false" type="Int" displayName="An
Input Integer" name="MyInt">
                  <Description>An Integer Parameter</Description>
            </InputParameter>
            <InputParameter array="false" type="Float"
displayName="An Input Float" name="MyFloat">
                  <Description>A Float Parameter</Description>
            </InputParameter>
            <InputParameter array="false" type="DateTime"
displayName="An Input Date" name="MyDate">
                  <Description>A Date Parameter in the form yyyy-MM-
dd</Description>
           </InputParameter>
            <InputParameter array="false" type="Boolean"
displayName="An Input Boolean" name="MyBoolean">
                  <Description>A Boolean (true-false)
Parameter</Description>
            </InputParameter>
            <InputParameter array="false" type="String"
displayName="An Input String" name="MyString">
                  <Description>A String Parameter</Description>
            </InputParameter>
            <InputParameter array="true" type="String"
displayName="inputStringArray" name="MyStringArray">
                  <Description>An Array of Strings</Description>
            </InputParameter>
            <InputParameter array="false" type="Int" displayName="An
Input Integer" name="arraySize">
                  <Description>An Integer Parameter</Description>
            </InputParameter>
            <InputParameter array="false" type="Int" displayName="An
Input Integer" name="index">
                  <Description>An Integer Parameter</Description>
            </InputParameter>
            <InputParameter array="false" type="String"
displayName="An Input String" name="temp">
                  <Description>A String Parameter</Description>
            </InputParameter>
            <OutputParameter array="false" type="Int"
displayName="An Output Integer" name="MyInt">
           <Description>The result of inputInt + 2</Description>
            </OutputParameter>
            <OutputParameter array="false" type="Float"
displayName="An Output Float" name="MyFloat">
                  <Description>The result of inputFloat +
1.11</Description>
```

```
            </OutputParameter>
           <OutputParameter array="false" type="DateTime"
displayName="An Output Date" name="MyDate">
                  <Description>The result of inputDate + 1
day</Description>
            </OutputParameter>
            <OutputParameter array="false" type="Boolean"
displayName="An Output Boolean" name="MyBoolean">
                  <Description>The negation of inputBoolean - true =
false and false = true</Description>
            </OutputParameter>
            <OutputParameter array="false" type="String"
displayName="An Output String" name="MyString">
                  <Description>The result of reversing inputString -
'abc' becomes 'cba'</Description>
            </OutputParameter>
            <OutputParameter array="true" type="String"
displayName="outputStringArray" name="MyStringArray">
                  <Description>The reverse array order of
inputStringArray - String1, String2, String3 becomes String3,
String2, String1</Description>
            </OutputParameter>
      </Method>
</ActivityDefinition>
```

User-Defined Functions

User-defined functions (UD functions) are defined as methods in code activities. They can be called through code activity or through a regular function call in any SAS Decision Services control language expression.

The new reserved Out parameter, “_RETURN_VALUE”, accommodates functions that return a value.

In and Out parameters are defined by using the same parameter name or type in both In and Out parameter lists. When called through a regular function call, the variable in the caller changes value as if it had been passed by reference.

The ANY type can be used for input parameters to code activities. As with other types, it can be an array or scalar. The UD function is responsible for checking the type of parameters.

Note: The ANY data type is not supported outside inputs to code activity.

UD functions are loaded based on function calls in active flows. All methods from an activity are loaded, even if only one is called. If there is a compile error in any method, all functions in the entire activity are rejected. Function name resolution during the compilation of methods is independent from the order of loading of activities.

Here are example activities. Each one calls a method in the other. Each one illustrates a function with a return value, and a function with an In or Out parameter.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
- <ActivityDefinition
     javaClassName="com.sas.analytics.ph.rt.act.code.CodeActivity"
  timeout="0" name="UDFunctionsCode1"
  xmlns="http://www.sas.com/xml/schema/sas-svcs/rtdm-1.1">
- <Method name="yes">
- <Body>
  <Expression>RETURN true;</Expression>
  </Body>
  <OutputParameter array="false" type="Boolean" name="_RETURN_VALUE" />
  </Method>
- <Method name="indirectCall">
- <Body>
  <Expression>RETURN UDFunctionsCode2.yes();</Expression>
  </Body>
  <OutputParameter array="false" type="Boolean" name="_RETURN_VALUE" />
  </Method>
  </ActivityDefinition>
```

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
- <ActivityDefinition
     javaClassName="com.sas.analytics.ph.rt.act.code.CodeActivity"
  timeout="0" name="UDFunctionsCode2"
  xmlns="http://www.sas.com/xml/schema/sas-svcs/rtdm-1.1">
- <Method name="yes">
- <Body>
  <Expression>RETURN true;</Expression>
  </Body>
  <OutputParameter array="false" type="Boolean" name="_RETURN_VALUE" />
  </Method>
- <Method name="indirectCall">
- <Body>
  <Expression>RETURN UDFunctionsCode1.yes();</Expression>
  </Body>
  <OutputParameter array="false" type="Boolean" name="_RETURN_VALUE" />
  </Method>
  </ActivityDefinition>
```

Functions are called as <Activity Name>.<Method Name>. Here are examples of a flow and sub-flow calling UD functions in every place possible:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <FlowDefinition eventName="AllTypes" timeout="2147483647"
  created="2008-11-18T15:25:23.097+00:00"
  modified="2008-11-18T15:25:23.097+00:00"
  lastModifiedBy="magres" name="UDFunctions1"
  xmlns="http://www.sas.com/xml/schema/sas-svcs/rtdm-1.1">
  <Description>Subflow (of UDFunctions) that calls
   UD functions</Description>
- <StartNode name="myStartNode">
- <Transition name="toReply">
  <Rule>UDFunctions1StartTr.yes()</Rule>
  <DestinationNodeName>myReplyNode</DestinationNodeName>
  </Transition>
  </StartNode>
  <ReplyNode name="myReplyNode" />
  <UIData />
  </FlowDefinition>

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <FlowDefinition eventName="AllTypes" timeout="2147483647"
  created="2008-11-18T15:25:23.097+00:00" modified="2008-11-18T15:25:23.097+00:00"
  lastModifiedBy="magres" name="UDFunctions"
  xmlns="http://www.sas.com/xml/schema/sas-svcs/rtdm-1.1">
  <Description>Flow that calls UD functions from a variety of places</Description>
  <ProcessVariable name="PVInt1" type="Int" array="false" />
- <StartNode name="myStartNode">
- <Transition name="toAssignment">
  <Rule>UDFunctionsStartTr.yes()</Rule>
  <DestinationNodeName>myAssignmentNode</DestinationNodeName>
  </Transition>
  </StartNode>
- <AssignmentNode name="myAssignmentNode">
- <Transition name="toSubflow">
  <Rule>UDFunctionsAssignmentTr.yes()</Rule>
  <DestinationNodeName>mySubFlowCallNode</DestinationNodeName>
  </Transition>
  <Assignments expr="UDFunctionsAssignmentRHS.yes()"
    processVariableName="PVInt1" />
  </AssignmentNode>
- <SubFlowCallNode eventName="UDFunctions1" name="mySubFlowCallNode">
- <Transition name="toCode">
  <Rule>UDFunctionsSubflowTr.yes()</Rule>
  <DestinationNodeName>myCodeNode</DestinationNodeName>
  </Transition>
  </SubFlowCallNode>
- <CodeNode name="myCodeNode">
- <Transition name="toActivityCall">
  <Rule>UDFunctionsCodeTr.yes()</Rule>
  <DestinationNodeName>myActivityNode</DestinationNodeName>
  </Transition>
  <Expression>PVInt1 = UDFunctionsCodeCode.yes();</Expression>
  </CodeNode>
- <ActivityMethodCallNode activityMethodName="method1"
  activityName="UDFunctionsAsActivity" name="myActivityNode">
- <Transition name="toReply">
  <Rule>UDFunctionsActivityTr.yes()</Rule>
  <DestinationNodeName>myReplyNode</DestinationNodeName>
  </Transition>
  </ActivityMethodCallNode>
  <ReplyNode name="myReplyNode" />
  <UIData />
  </FlowDefinition>
```

The easiest way to package UD functions is to group related functions together as multiple methods in a single code activity.

Guidelines for Creating Activities

Date Time Formats That Are Supported by SAS Decision Services

SAS Decision Services I/O recognizes SAS DATETIME rather than SAS DATE.

Note: A SAS DATE value is a value that represents the number of days between January 1, 1960, and a specified date. A SAS DATETIME value is a value that represents the number of seconds between January 1, 1960, and an hour/minute/second within a specified date.

SAS data sets can store dates as DATETIME or DATE. SAS Decision Services supports a single datetime data type. When datetime values are passed from SAS Decision Services to SAS, they are always converted into SAS DATETIME values. When these values are used to insert or update a value in a SAS data set, they update the value as the number of seconds from January 1, 1960, rather than the number of days. If the data set column is then viewed with a DATE format for that column, then the value is displayed incorrectly. Always use a DATETIME format to view such columns.

Boolean Values

Within custom SAS activities, Boolean values must be represented as the numerics 0 and 1, as opposed to `True` and `False`.

General I/O Write and SAS Data Sets

SAS data sets do not support concurrent updates. Therefore, locking errors can occur if you try to use General I/O to insert records into a SAS data set or to update records in a SAS data set. If concurrent writes are required, then use a database table.

Note: This action is for Write actions only; concurrent Read actions are fully supported.

If a data set is opened in an interactive SAS session while SAS Decision Services is reading the data set, locking errors occur. The errors occur because SAS locks the file when it is opened. It is recommended that all other SAS data sets be closed in an interactive SAS session while SAS Decision Services is using the SAS data set.

6

Integration

Web Service Integration

Overview

External applications see the SAS Decision Services engine server as a Web service endpoint. They request decisions by sending Web service requests to SAS Decision Services. When the endpoint is triggered by a Simple Object Access Protocol (SOAP) event request, the Web service maps the incoming request to a SAS Decision Services event object. It then passes it to the run-time engine for processing. After the run-time engine has completed its processing, a SOAP response is serialized back to the invoking client.

One-way event operations are also supported, but they do not follow the common request and response message exchange pattern that is described above. In this case, a client sends a request and does not expect a response. Specifically, SAS Decision Services supports SOAP document-style encoding, also known as *document-literal* or *message-style* encoding. Of the three most popular SOAP encoding styles, SOAP RPC, SOAP RPC-literal, and document-literal, the document-literal style has the least overhead and highest performance.

The variables in the SOAP messages are accessed by name, and the order of declaration is not significant. In particular, the variables in the SOAP messages are independent of the order of the variables that are defined in the request and the reply message sections of the event definition. Client applications should not rely on reply variables being returned in any particular order.

Web Service Definition Language

Retrieve a WSDL File

You can retrieve the Web Service Definition Language (WSDL) file for a given SAS Decision Services event by invoking an export process as follows:

1. Open SAS Management Console.

2. On the **Folders** tab, select **System** ▶ **Applications** ▶ **SAS Decision Services** ▶ **Decision Services 5.5**.

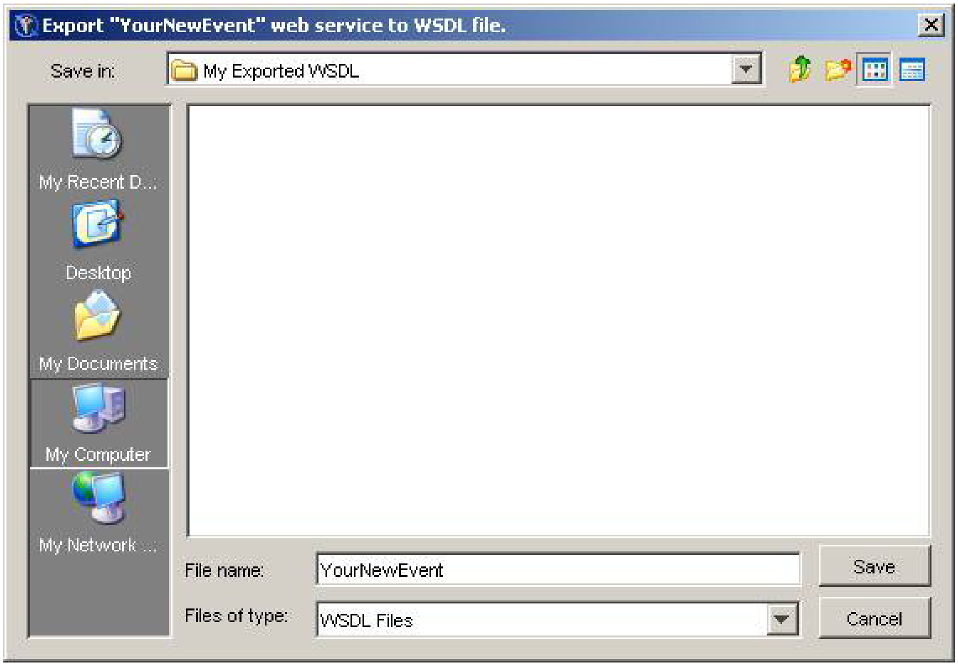
3. Navigate to the SAS Decision Services repository for which you want to generate a WSDL file.

4. Right-click the event in the repository and click **Export WSDL**.

5 Modify the default address for your environment. A sample address is: **`http://`** ***`localhost`*** **`:9086/RTDM/Event`**. The address is determined during the installation of your software.

6 Navigate to a location to store and name the WSDL file.

Display 6.1 *WSDL File*



Export "YourNewEvent" web service to WSDL file.

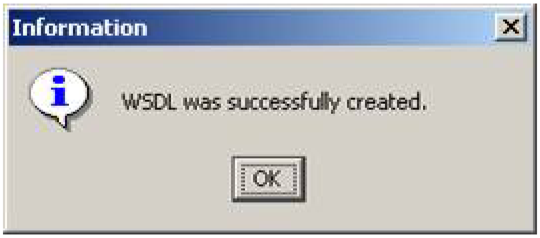
Save in: My Exported WSDL

File name: YourNewEvent

Files of type: WSDL Files

7 Click **Save**.

Display 6.2 *WSDL Created*

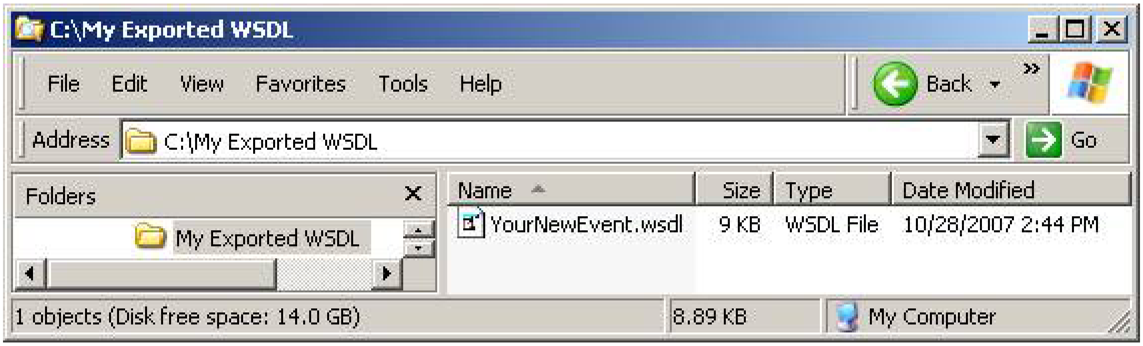


Information

WSDL was successfully created.

Verify that the new WSDL exists by browsing the directory to locate the file.

Display 6.3 Exported WSDL



C:\My Exported WSDL

Name	Size	Type	Date Modified
YourNewEvent.wsdl	9 KB	WSDL File	10/28/2007 2:44 PM

1 objects (Disk free space: 14.0 GB) 8.89 KB

Sample Web Service Request

After creating an event and mapping that event to a decision flow, you can deploy the flow to a running instance of the SAS Decision Services engine server. After the decision flow is activated, the event can be invoked by a Web service client. Here is a sample instance of a SOAP request that calls an event named "CustomerCall":

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance>"
    <env:Header/>
    <env:Body>
         <rdm:Event xmlns:rdm="http://www.sas.com/xml/analytics/rdm-1.1"
name="CustomerCall">
                <rdm:Header>
                      <rdm:Identity>John Smith</rdm:Identity>
                  <rdm:ClientTimeZoneID>America/New_York</ClientTimeZoneID>
                </rdm:Header>
               <rdm:Body>
                      <rdm:Data name="CustomerID>"
                            <rdm:String>
                                 <rdm:Val>001</rdm:Val>
                            </rdm:String>
                      </rdm:Data>
                      <rdm:Data name="Amount">
                            <rdm:Float>
                                 <rdm:Val>25000.0</rdm:Val>
                            </rdm:Float>
                      </rdm:Data>
                      <rdm:Data name="Mood">
                            <rdm:String>
```

```
                                <rdm:Val>Good</rdm:Val>
                           </rdm:String>
                     </rdm:Data>
                     <rdm:Data name="SigEvent">
                           <rdm:String>
                                <rdm:Val>NewBaby</rdm:Val>
                           </rdm:String>
                     </rdm:Data>
               </rdm:Body>
         </rdm:Event>
   </env:Body>
</env:Envelope>
```

ClientTimeZoneID is a required tag in the SAS Decision Services header. Time zone names from the public domain time zone (TZ) database are accepted. The following Web site lists time zone information from the TZ database: **`http://home.tiscali.nl/~t876506/TZworld.html#nam`**.

Every Web service stack has client tools that can be used to generate both stubs and helper classes that call particular Web services. These toolsets take the desired Web service's WSDL file as input and generate the stubs and helper classes as output. Clients can be plain Java or .Net applications or, in a J2EE setting, they can be J2EE application clients or J2EE Web applications themselves.

Integration with SAS Model Manager

About SAS Model Manager

SAS Model Manager, licensed separately, can be integrated with SAS Decision Services to provide an end-to-end solution for managing and deploying analytical models into real-time operational environments.

SAS Model Manager 12.1 is the required version for use with SAS Decision Services.

See the SAS Model Manager documentation for information. This section describes the integration and interoperability between SAS Decision Services and SAS Model Manager.

SAS Decision Services is shipped with an activity called Scoring. When SAS Decision Services is integrated with SAS Model Manager, the Scoring activity provides several features:

- In the development environment, the Scoring activity enables a user to choose any of the scoring projects that have been published to SAS Decision Services by SAS Model Manager. The Scoring activity can be added to a decision flow in multiple places, allowing multiple models to be included in a single decision flow.
- In the production environment, whenever a decision flow that includes Scoring activities is executed, those activities execute the referenced score code.
- Indirect reference to models. SAS Decision Services does not reference models directly. Rather, it references scoring projects, which in turn contain models. At run time, the champion model within the scoring project is used. This indirection allows new versions of models to be deployed without redeploying the decision flows that reference them.

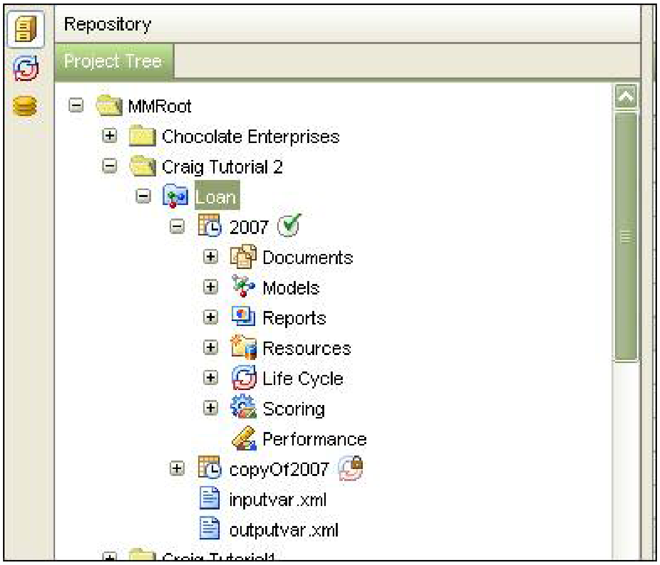
Publish a Scoring Project

The installation and configuration of SAS Model Manager creates the directory structure in SAS Management Console for housing scoring projects. The default location (on the **Folders** tab in SAS Management Console) is **Shared Data** ▶ **Model Manager** ▶ **Publish** ▶ **Projects**.

To publish a scoring project to SAS Decision Services:

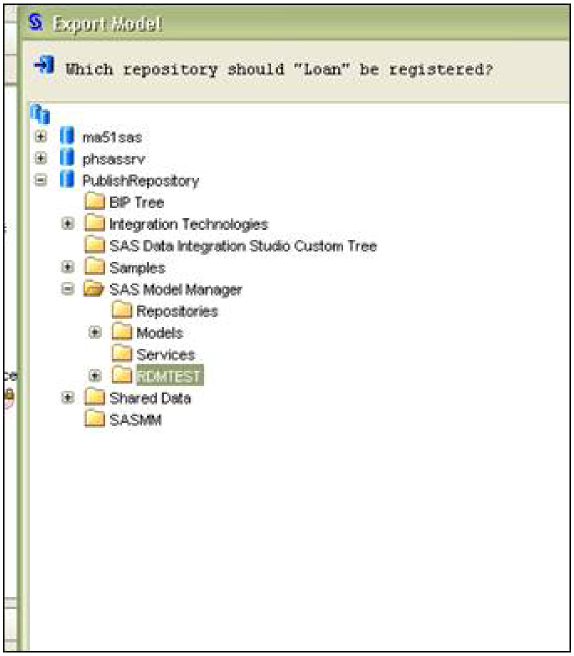
1 Open Model Manager and navigate to the scoring project that you want to publish. The example uses the Loan project:

Display 6.4 *Model Manager*



2 Export the project to the publish folder (RDMTEST) in the BIP tree. Note that two levels of export are supported by SAS Model Manager, project-level and model-level. Always export from the project-level.

Display 6.5 *Export Model*



Which repository should "Loan" be registered?

The project information and files have now been transferred to the Metadata Server, and the project is ready to be extracted by the Model Update Service. The project information includes the score code, input, and output variables, which are needed for scoring. Also included is information about the project and champion model that can be presented in SAS Customer Intelligence Studio 5.4.

Model Update Publishing

Model Update Publishing distributes scoring projects that are published by SAS Model Manager to the DataFlux Federation Server in a cluster. To publish models to the DataFlux Federation Server for use by SAS Decision Services, right-click on the content repository in the SAS Management Console and click **Publish**.

As part of the configuration process, a catalog name was set for the models. To find the value of this property:

1. From the **Plug-ins** tab of SAS Management Console, navigate to **Application Management** ▶ **Configuration Manager** ▶ **SAS Application Infrastructure**.

2. Right-click **Decision Services Design Mid-Tier 5.5**, and click **Properties**.

3. Click the **Advanced** tab.

   Record the property value for the property name policydesign.model.ds2.catalog.

4. Repeat this process in SAS Application Infrastructure for SAS Decision Services Engine Server 5.5, and record the property value for the property name policyengine.model.ds2.catalog.

Each property value must match a library name in the Data Library Manager plug-in, and the libraries must be marked as pre-assigned. To verify the library name and assignment:

1. From the **Plug-ins** tab of SAS Management Console, navigate to **Environment Management** ▶ **Data Library Manager** ▶ **Libraries**.

2 Right-click the library with the name that matches the property value that you recorded in step 3 or 4 earlier, and select **Properties**.

3 Click the **Options** tab, and then **Advanced Options**. If the library was pre-assigned, there is a check in the **Library is Pre-Assigned** check box.

4 Repeat these steps for the other property name.

If the stored process server was running before the creation of this library, the stored process server must be stopped and restarted in order to pick up the library name. The default schema for the credentials that are used in the library is the schema in which the model DS2 package tables are stored.

Best Practices

- One SAS Metadata Repository folder for publishing models should be created for each development, test, and production SAS Decision Services environment in your deployment.
- A scoring project should be published to the development folder first and tested in the SAS Decision Services development environment.
- When a decision flow that references the scoring project is promoted to a test or production environment, the scoring project should then be published to the test or production folder.
- Using this practice, the same testing, approval, and promotion policies that are applied to decision flows can be applied to scoring projects.

Integration with SAS Data Surveyor for Clickstream Data

About SAS Data Surveyor for Clickstream Data

The term *clickstream* describes the data that is collected from users as they access Web sites through various electronic devices. Clickstream data includes the stream of user activity that is stored in a log. Clickstream data can be collected and stored in a variety of ways. SAS Data Surveyor for Clickstream Data enables you to process this data and produce meaningful results.

Integration of SAS Data Surveyor for Clickstream Data with SAS Decision Services allows for real-time campaign content to be presented to the Web site visitor. The real-time content is based on information that is specific to the visitor’s session. Any subsequent activity that the user takes on the presented content is tracked. This tracking can help with determining the success of campaigns and analyzing customers responses to different types of content that are presented within a campaign.

Real-Time Behavior Tracking and Analysis

The SAS page tag functionality passes session information through an asynchronous request to the SAS Decision Services Web service, which responds with a targeted response. The response contains information about a treatment that should be displayed on the current Web site. For example, the treatment can identify an image that is contained in the content management system. The SAS page tag functionality then updates the Web page source to display the appropriate content. When the customer clicks on this treatment (which is typically a link), information about the campaign that generated the treatment is recorded in the SAS page tag log. This information is later processed as part of the extract-transform-load (ETL) process.

See *SAS Data Surveyor for Clickstream Data User's Guide* for information about configuration and use of the combined functionality provided by SAS Data Surveyor for Clickstream Data and SAS Decision Services.

Integration with SAS Rule Studio

Overview

Rule sets that are created with SAS Rule Studio can be used in decision flows. To include a rule set in a decision flow, first convert the rule set to DS2 and then publish it as a SAS activity. For more information about SAS activities, see "SAS Activities" on page 65.

Publishing Rule Sets as SAS Activities

Publishing a custom SAS activity is a three-step process:

1. Write DS2 package code that implements the logic of your activity methods.
2. Publish your DS2 package to DataFlux Federation Server using PROC DS2.
3. Publish activity metadata (accomplished with the Customer Intelligence plug-in by creating a new SAS process).

When publishing a rule set as a SAS activity, SAS Rules Manager generates the logic of the activity for you, eliminating step 1 above. SAS Rule Studio stores this logic as XML in a SAS Metadata Repository object, which must be converted to DS2 before it can be used by SAS Decision Services. SAS Decision Services includes a command-line utility that performs this conversion for you.

To convert one or more SAS Rule Studio rule sets to DS2, run `GenerateRulesSet`. This command produces .sas files that contain your new DS2 activity packages. Use these DS2 packages to complete the remaining SAS activity publishing steps. Here is the syntax for `GenerateRulesSet`.

```
GenerateRulesSet [-?] [-all] [-debug] [-domain <domain>] [-host
       <hostname>] [-log <log-file>] [-nolog] -outdir <outdir> [-password
       <password>] [-port <port>] [-print] [-profile <profile>]
       [-repository <repository>] [-resource <resource repository>]
       [-source <source>] [-sourceFolder <sourceFolder>] [-user <userid>]
```

Options	Descriptions
-?,-help	Prints help information.
-all	(Optional) Indicates whether to create DS2 packages for all SAS Business Rules Manager business rule flows in the SAS Metadata Repository. Either -source, -sourceFolder, or -all must be used.
-debug	Prints debugging information.
-domain <*domain*>	Provides user authentication domain information.
-host <*hostname*>	Metadata server host. Required if -profile is not set.
-log <*log-file*>	Log file or directory.
-nolog	Disable log file.
-outdir <*outdir*>	(Required) The name of the output directory to which the rules set DS2 packages are written.
-password <*password*>	User login password. Required if -profile is not set or if the profile does not contain connection credentials.
-port <*port*>	Metadata server port. Required if -profile is not set.
-print	(Optional) Prints the names of the available Business Rule Flows. Used in conjunction with either the -all or the -sourceFolder options.
-profile <*profile*>	Metadata server connection profile. Can be used instead of the -host, -port, -user, and -password options.

-repository <*repository*>	(Optional) The name of the metadata repository. If this option is missing, the default name of "Foundation" is used.
-resource <*resource repository*>	(Optional) The resource to use for storing the DS2 package and the Decision Services repository in which it is located. If this option is missing, a place holder is put into the DS2 package code.
-source <*source*>	(Optional) The name of the SAS Business Rules Manager business rule flow, for which DS2 packages are generated.
-sourceFolder <*sourceFolder*>	(Optional) The name of the SMR Folder that contains the Rules Manager Business Rule Flows to generate DS2 packages for. Either the -source, -sourceFolder, or -all options must be used.
-user <*userid*>	User login identity. Required if -profile is not set or if the profile does not contain connection credentials.

7

Installation

Overview

Before installing SAS Decision Services, work with your on-site SAS support personnel to determine the hardware, network, and software topology that is required to meet your throughput and response time needs.

Choosing Environments

At a minimum, install one development and one production environment. You can install one or more test environments, depending on your organization's testing policies. Decision flows can be unit tested in the development environment. A test environment is used to test decision flows in an environment that is similar to production. The test and production environments have only a few differences:

- The test environment is not connected to live channels or customer-facing systems.
- The production environment might have more hardware and network resources allocated to it.

The development environment is typically not clustered. The production environment might use a clustered middle tier, compute tier, database tier, and DataFlux Federation Server tier.

Third-Party Software Components

Web Application Server

The Web application server is the container for SAS Decision Services Web applications.

The following third-party servers are supported:

- JBoss Application Server
- IBM WebSphere Application Server
- Oracle WebLogic Server

For more information about the supported versions of these products and supported platforms, see the SAS third-party Web page at **`http://support.sas.com/resources/thirdpartysupport/v93`**.

Configuration instructions vary depending on the Web application server that is installed at your site. For configuration instructions that pertain to the topics that are discussed in this chapter, see the following third-party vendor Web sites:

- JBoss Application Server: **`http://www.jboss.org/docs`**
- IBM WebSphere Application Server: **`http://www.ibm.com/support/documentation/us/en`**
- Oracle WebLogic Server: **`http://www.oracle.com/technology/documentation/index.html`**

If your deployment uses JBoss, and you want to configure the middle-tier environment manually, then configure separate Web application server instances for the SAS Decision Services Web applications. Do not deploy the SAS Decision Services Web applications to Web application server instances that are used for other Web applications. Likewise, do not deploy other Web applications to the Web application server instances that are used for the SAS Decision Services Web applications.

If your deployment uses IBM WebSphere Application Server, and you want to configure the middle-tier environment manually, then configure a separate cell for the SAS Decision Services Web applications. The SAS Decision Services Web applications use resources that are configured at the cell level. Configuring a separate cell avoids interference between the SAS Decision Services Web applications and other Web applications.

If your deployment uses Oracle WebLogic Server, and you want to configure the middle-tier environment manually, then configure a separate domain for the SAS Decision Services Web applications. The SAS Decision Services Web applications use resources that are configured at the domain level. Configuring a separate domain avoids interference between the SAS Decision Services Web applications and other Web applications.

Java Development Kit

If you are using JBoss or WebLogic Server, you must install a Java Development Kit (JDK) for compiling the SAS Web applications. WebSphere Application Server is shipped with a JDK. For more information about the supported versions of the JDK, see the SAS third-party Web page at **`http://support.sas.com/resources/thirdpartysupport/v93`**.

DataFlux Authentication Server

The DataFlux Authentication Server is part of the DataFlux Data Management Platform. The platform provides centralized data access and data analysis for the data that is stored in DataFlux servers and databases across your enterprise.

The DataFlux Authentication Server is required in all deployments that include a DataFlux Federation Server.

For more information, see the *DataFlux Authentication Server Administrator’s Guide* that is located in the **`doc`** folder of your authentication server installation directory.

DataFlux Standard Federation Server

The DataFlux Federation Server is a compute server that executes SAS Decision Services activities that are written in the DS2 programming language.

For more information, see the *DataFlux Federation Server Administrator's Guide* that is located in the `doc` folder of your DataFlux Federation Server installation directory.

Dependent SAS Products

SAS Model Manager

SAS Model Manager enables you to manage scoring project life cycles and to publish analytical scoring models for use by SAS Decision Services. See "Integration with SAS Model Manager" on page 93, for more information.

SAS BI Web Services for SAS 9.3

SAS BI Web services for SAS 9.3 enables you to select a set of stored processes in SAS Management Console and use the Web Service Maker to deploy them as Web services. The Web Service Maker generates a new Web service that contains one operation for each stored process that you selected. For more information about developing Web services, see the *SAS BI Web Services Developer's Guide*.

To invoke a SAS BI Web service from SAS Decision Services, include a Web service activity in your decision flow. SAS BI Web services are useful if you want to execute multiple DATA or PROC steps, or if you want to use SAS macro code. However, keep in mind that these code constructs carry significant performance penalties.

Best Practices for SAS Decision Services Deployment Scenarios

Overview

Decision services consist of processing steps (called activities) and conditional control logic. The conditional logic determines which activities are executed and in what order. The path of execution through a decision service is typically influenced by the input data and by the results of each processing step. The response time for a single execution of a decision service is the sum of the latencies of the processing steps along this path of execution.

Because paths of execution are data dependent, a single decision service might exhibit a range of latencies. Furthermore, multiple heterogeneous decision services can be deployed at the same time, each consuming a portion of the available computational resources. It is often impractical to attempt to anticipate all possible combinations of data and their influences on performance, making hardware capacity planning a challenge.

Therefore, it is a good practice to create a baseline system to deploy your decision services into it, measure performance against historical data, and extrapolate the results to create a hardware plan that meets your throughput and latency requirements.

The following sections explain how to create an appropriate baseline SAS Decision Services environment. The SAS Decision Services environment can be used to collect the data necessary to plan the production hardware capacity that is required for your own unique set of decision services.

Component	Used By	Performance Critical?	Tier
SAS Management Console	Design and Run time	No	Client
SAS Metadata Server	Design and Run time	No	SAS

Third-party Database Management System	Design and Run time	Yes	Database (DBMS)
SAS Decision Services 5.5 Design Server	Design	No	Middle
SAS Decision Services 5.5 Engine	Run time	Yes	Middle
DataFlux Federation Server 2.1.3	Design and Run time	Yes	Compute
DataFlux Authentication Server 2.1.3	Design and Run time	No	Compute
DataFlux Data Management Studio 2.1.3	Design and Run time	No	Client
SAS Object Spawner	Run time	No	SAS

Best Practices for SAS Decision Services Performance

- SAS Decision Services has a design environment and a run-time environment. The design environment is used for developing, modifying, and functional testing of decision services. The run-time environment is used for production. It can also be used for integration or performance testing. High performance, measured in terms of throughput and latency, is typically critical for run-time environments and less important for design environments.
- The following components are critical to run-time performance:
  - engine
  - DataFlux Federation Server
  - database management system

- In the run-time environment, the engine, DataFlux Federation Server, and database instances should be installed on dedicated hardware. Service levels cannot be guaranteed if external software is allowed to consume resources.
- The DataFlux Federation Server has approximately twice the throughput capability as the engine. Therefore, an optimized deployment, for CPU-bound processing, includes one of the following options:
  - two engine servers per DataFlux Federation Server
  - assigning more powerful hardware to the engines than to the DataFlux Federation Server
- The numbers of servers that are allocated to the middle, SAS, and database tiers should be proportional to your throughput and latency requirements.
- Open Metadata Supervisor, SAS Management Console, and the object spawner have a minimal impact on performance because they are not directly involved in transaction processing.
- The DataFlux Federation Server cannot be collocated with SAS 9.3 because of conflicts between it and Framework Data Server. Otherwise, tiers can be collocated in deployments where optimum performance is not required.

Deployment Scenarios

Easy Button

An "easy button" deployment is a deployment where the default settings, where available, are used during the installation and configuration process. This results in a design-time system and a run-time system. Easy button deployments are suitable for decision service design and functional testing, but are appropriate only for production use in cases where high performance and high availability are not required.

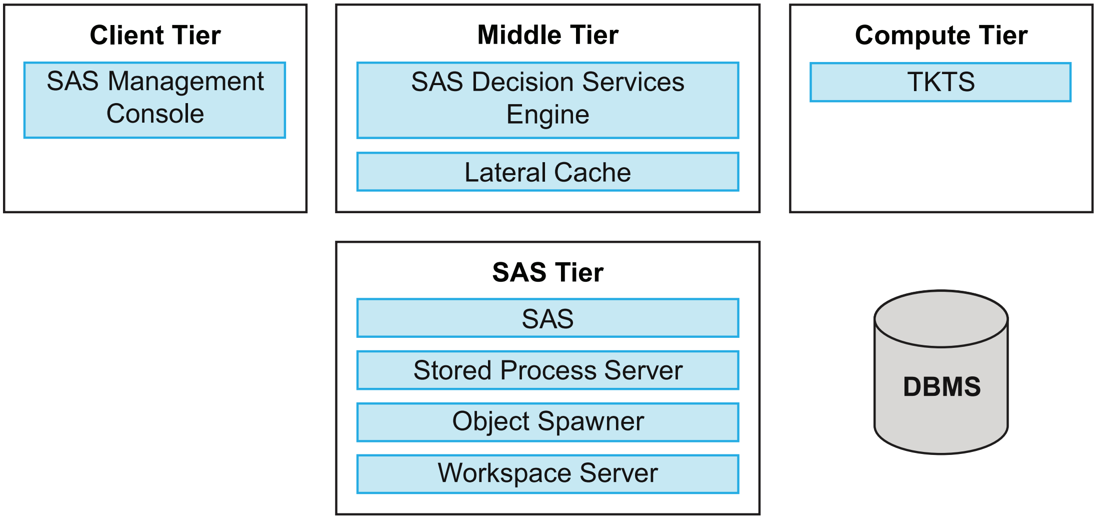
The design-time system contains a design server for creating and modifying decision services. It contains many of the same software components as a production system, in order to enable functional testing of decision services. A major difference between a design environment and a production environment is that a production deployment typically includes load-balanced, clustered engine servers and multiple DataFlux Federation Server instances for scalability and high availability.

Production Deployments

SAS Decision Services production deployments consist of the following major components:

- SAS Decision Services engine server cluster
- one or more DataFlux Federation (TKTS) Server and DataFlux Authentication Server pairs
- SAS Management Console plug-in, for centralized control and monitoring
- third-party load balancer
- third-party database management system
- SAS Stored Process Server for the execution of BI Web services

Figure 7.1 *Production Deployment*



Here are examples of the factors that actual hardware capacity planning depends on:

- peak transaction load
- maximum latency requirements

- minimum throughput requirements
- business logic complexity
- analytic complexity
- size of request and response messages
- amount and frequency of disk I/O
- external system dependencies, such as external Web service calls made by a decision service

Scenario: Complex Business Logic and Light Analytics

A typical SAS Decision Services scenario might include business logic combined with one or two high-performance predictive models that generate scores, such as propensity or risk. For the purposes of this scenario, assume that all required data is passed in to the decision service through the event. Therefore, no database I/O occurs. In such a scenario, processing is approximately evenly divided between the business logic and the analytics. In general, the business logic executes in the engine middle tier and the analytics execute inside DataFlux Federation Server.

Because a DataFlux Federation Server executes with approximately twice the throughput of the engine middle tier, a baseline topology might include a 16-core middle tier server and an 8-core DataFlux Federation Server. Alternatively, two engine servers could be allocated per DataFlux Federation Server, if all servers are equally powerful.

The baseline topology hardware should be multiplied until latency and throughput requirements are achieved.

A hardware failover capability requires at least two servers per tier. All DataFlux Federation Servers should have access to a common clustered database management system. This database should be clustered to support failover. A common database should be used to allow all DS2 activity packages to be accessed by all DataFlux Federation Servers. Using servers of equal capacity for this scenario, a system capable of hardware failover would have four middle tier servers, two DataFlux Federation Servers, and a database server cluster.

Although data I/O was not included in this scenario, a database management system is nevertheless required to store the activities, which are persisted in the database as DS2 packages.

Scenario: Complex Business Logic and Complex Analytics

Another typical SAS Decision Services scenario includes both complex business logic and complex analytics, where three or more predictive models, or one or more very complex models, are used. In this scenario, the analytics require more processing cycles than the business logic.

Because a DataFlux Federation Server executes at approximately twice the speed of the engine middle tier and is doing twice as much work, a baseline topology might include a 16-core middle tier server and a 16-core DataFlux Federation Server.

The baseline topology should be multiplied until latency and throughput requirements are achieved.

A hardware failover capability requires at least two servers per tier. As mentioned earlier, all DataFlux Federation Servers should have access to a common clustered database management system. For this scenario, a typical system capable of hardware failover would have two middle tier servers, two DataFlux Federation Servers, and a database server cluster.

Heterogeneous Decision Service Considerations

In reality, most deployments include a mix of variations on the scenarios described earlier. To determine the proportion of processing cycles consumed by business logic versus analytics, the cumulative effects of each decision service must be considered, as well as the relative frequencies of the events bound to each. When measuring your baseline system using historical data, record CPU utilization for each server at several points during the simulation run. The results indicate whether the processing load is balanced, and where adjustments to hardware resources must be made.

Database I/O Considerations

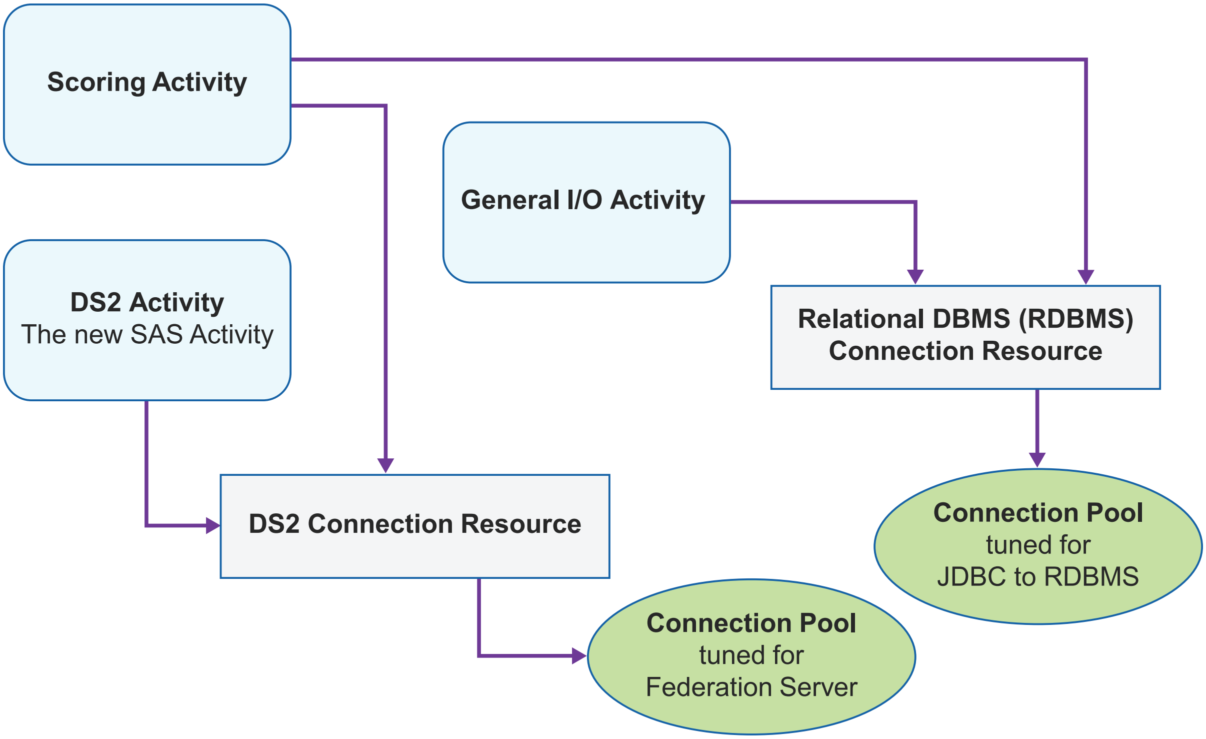
Disk I/O is by far the most expensive operation that real-time systems perform. Accessing data on disk is typically 10**5 times more expensive than accessing data in memory. Therefore, database hardware capacity planning and performance tuning are critical to the performance of decision services that read from or write to disk. Contact your database management system vendor for guidance.

Connection and Statement Pool Tuning

When you are using 8-way to 16-way blade servers, the optimum connection pool and prepared statement pool sizes are both between 16 and 24 inclusive. A DataFlux Federation Server allocates a thread per JDBC Connection, so pool size has a direct effect on performance.

Your individual hardware might perform differently, so you might want to experiment with different pool sizes to achieve optimum performance. Pool tuning controls are contained by the JDBC System Resource, which can be accessed through SAS Management Console. If you are unfamiliar with database performance tuning principals, consult your DBA or database vendor for assistance.

Figure 7.2 Activities, Resources, and Connection Pools



Configuring SAS Decision Services

For more information about configuring SAS Decision Services, see the *SAS Decision Services 5.5 Configuration Guide*.

Deploying and Starting SAS Decision Services Web Applications in a Cluster

The SAS Deployment Wizard deploys SAS Decision Services Web applications to the Web application server. However, you can also deploy Web applications manually from the Web application server. The Web applications are in the **`SAS-config-dir\Lev1\Web\Staging`** directory.

There is no required start-up order for deploying the Web applications to JBoss or the WebLogic Server. Although you can deploy and start the Web applications in any order, it is recommended that you follow the sequence that is used for the WebSphere Application Server. For a WebSphere Application Server, the sequence for starting the Web applications is important. The following recommended sequence can be used directly as the number to enter in the **Startup order** field for a WebSphere Application Server.

1. SAS Decision Services Design Middle Tier (sas.rtdm.designserver5.5.ear)

2. SAS Decision Services Engine Server (sas.rtdm.engine.webservice5.5.ear)

Rebuilding SAS Decision Services Design and Engine Server Web Applications

Rebuild Web Applications

Rebuild Web Applications in SAS Deployment Manager enables you to rebuild one or more Web applications. The rebuild process updates two directories for each rebuilt Web application:

- `SAS-config-dir\Lev1\Web\Staging:` An EAR file for each rebuilt Web application is placed in this directory. The approximate size of the collection of EAR files for EBI is 2 GB.
- `SAS-config-dir\Lev1\Web\Staging\exploded`: An exploded version of each rebuilt Web application is placed in this directory. The approximate size of the entire exploded directory is 2 GB. The size is similar to the size of all the EAR files in the Staging directory.

  Note: You can delete any unwanted directories in the exploded directory to save disk space.

To rebuild one or more Web applications:

Note: The Web application server can be running or stopped.

- For WebLogic Server, the administration server and node manager can be running or stopped.
- For WebSphere Application Server, the deployment manager and node agent can be running or stopped.

1. Make sure that the SAS Metadata Server is running.
2. Start the SAS Deployment Manager.
3. Select **Rebuild Web Applications** and click **Next**.

4 Specify the configuration directory and the level (for example, Lev1) on the Select Configuration Directory/Level window. Click **Next**.

5 Enter the user ID and password for an unrestricted administrative user (for example, sasadm@saspw) on the Specify Connection Information page. Click **Next**.

6 Select the check boxes for the Web applications that you want to rebuild. Click **Next**.

7 Review the Summary page and click **Start**. The SAS Deployment Manager builds the EAR files for the selected applications.

Names of the SAS Decision Services Web Applications and EAR Files

The files for the SAS Web applications are stored in the following directories:

- `SAS-config-dir\Lev1\Web\Staging`
- `SAS-config-dir\Lev1\Web\Staging\exploded`

When the SAS Deployment Manager is used to rebuild a SAS Decision Services Web application, the files for the Web application in the previous directories are overwritten. The following table identifies the product configuration name that is used in the SAS Deployment Manager for the SAS Decision Services Web applications. Use this table to understand which Web applications and EAR files are updated when a product configuration is selected in the SAS Deployment Manager.

Product Configuration	EAR File
SAS Decision Services Design Middle Tier 5.5	(sas.rtdm.designserver5.5.ear)
SAS Decision Services Engine Server 5.5	(sas.rtdm.engine.webservice5.5.ear)

Redeploying the SAS Web Applications

For more information about redeploying the SAS Web applications, see the “Administering SAS Web Applications” chapter of the *SAS 9.3 Intelligence Platform Middle-Tier Administrator’s Guide*.

Rebuilding the Web Application Server Configurations

For more information about rebuilding the Web application server configurations, see the “SAS Configuration Scripting Tools” chapter of the *SAS 9.3 Intelligence Platform Middle-Tier Administrator’s Guide*.

Post-Installation Reconfiguration

Engine and Design Server Reconfiguration

You must unconfigure and then reconfigure the SAS Decision Services engine and design servers using the SAS Deployment Wizard.

8

Migration

Overview

SAS Decision Services 5.5 contains significant architectural advances over earlier releases. With these advances, some metadata and code objects were changed, some were replaced, and some were left unchanged:

- Single DATA step activity SAS code has been converted from DATA step to DS2.
- SAS Connection system resources were replaced by JDBC Connection system resources.

- Single DATA step metadata structure is unchanged.
- Multiple DATA step activities are no longer supported. However, if you have custom multiple DATA step activities, you can host the code in a stored process and call it through BI Web services. This strategy allows PROC and macro code to be used. However, in general, such code is not appropriate for real-time processing.
- Global variables are unchanged.
- Flows are unchanged, unless they contain a reference to a multiple DATA step activity. In this case, the flow must be modified to either remove the reference or to reference a BI Web service instead.

SAS Decision Services 5.5 is shipped with DS2 versions of all activities. During the installation and configuration process, the SAS Migration Utility modifies each custom single DATA step activity. As a result, the SAS Connection resource reference is replaced with a reference to the appropriate JDBC Connection system resource. Also, during the installation and configuration process, another utility converts to DS2 any custom single DATA step activity code that is written in a DATA step. These utilities can be run independently after your system is configured, in order to migrate any additional artifacts that predate SAS Decision Services 5.5 that you might have.

Migration from the third maintenance release for SAS Real-Time Decision Manager 5.3 to SAS Decision Services 5.5, as well as from SAS Real-Time Decision Manager 5.4 to SAS Decision Services 5.5 is supported.

Before you upgrade the third maintenance release for SAS Real-Time Decision Manager 5.3 or 5.4, it is recommended that you export your user-defined flows, events, resources, and activities manually before applying the upgrade. To export these resources:

1 Launch SAS Management Console. Be sure to select the metadata profile of a user who has Edit permissions in the design and production repositories.

2 Navigate to each user-defined repository folder in **`System\Applications\SAS Real-Time Decision Manager\SAS Real-Time Decision Manager 5.3M3 or 5.4`**.

3 Right-click on the repository folder level, and select the export package menu item. Repeat this step for each folder repository.

4 Make a backup copy.

Migration from SAS Real-Time Decision Manager 5.3

To migrate from the third maintenance release for SAS Real-Time Decision Manager 5.3 to SAS Decision Services 5.5:

1 If you are performing a manual migration, run the Artifact Migration utility to migrate your third maintenance release 5.3 metadata artifacts to 5.5 format. For more information, see the ArtifactMigration program in “SAS Decision Services Utilities” on page 124. In the case of an automatic migration, this step is done for you by the SAS Deployment Wizard.

2 Convert your third maintenance release for SAS Real-Time Decision Manager 5.3 single DATA step SAS code to SAS Real-Time Decision Manager 5.4 format. To do that, follow the instructions in “SDS Server” on page 156.

3 Run the Migrate utility to convert your SAS Real-Time Decision Manager 5.4 formatted single DATA step SAS code to Decision Services 5.5 format. For more information, see the Migrate program in “SAS Decision Services Utilities” on page 124 as well as “Migrate Single DATA Step SAS Code” on page 123.

4 Convert your third maintenance release for SAS Real-Time Decision Manager 5.3 multiple DATA step SAS code to SAS Real-Time Decision Manager 5.4 format. To do that, follow the instructions in “MDS Server” on page 167.

5 Convert your SAS Real-Time Decision Manager 5.4 formatted multiple DATA step SAS code into BI Web service activities. To do that, follow the instructions in “Migrate Multiple DATA Step Code” on page 128.

Migration from Real-Time Decision Manager 5.4

To migrate from SAS Real-Time Decision Manager 5.4 to SAS Decision Services 5.5:

1 If you are performing a manual migration, run the Artifact Migration utility to migrate your SAS Real-Time Decision Manager 5.4 metadata artifacts to SAS Decision Services 5.5 format. For more information, see the ArtifactMigration program in “SAS Decision Services Utilities” on page 124. In the case of an automatic migration, this step is done for you by the SAS Deployment Wizard.

2 Run the Migrate utility to convert your SAS Real-Time Decision Manager 5.4 single DATA step SAS code to SAS Decision Services 5.5 format. For more information, see the Migrate program in “SAS Decision Services Utilities” on page 124 as well as “Migrate Single DATA Step SAS Code” on page 123.

3 Convert your SAS Real-Time Decision Manager 5.4 multiple DATA step SAS code into BI Web service activities by following the instructions in “Migrate Multiple DATA Step Code” on page 128.

Migration from SAS Decision Services 5.5 to 5.5M1

In order to take advantage of the most recent enhancements provided by the SAS Decision Services 5.5 maintenance release, take the following steps on your existing SAS Decision Services 5.5 deployment:

1 Install and configure the latest release of the DataFlux Federation Server, Authentication Server, Data Management Studio, and Federation Server Manager. For more information, see the *DataFlux Federation Server Administrator’s Guide* and the *DataFlux Authentication Server Administrator’s Guide*.

2 Install the maintenance release of SAS Decision Services 5.5. For instructions, see the *SAS Deployment Wizard and SAS Deployment Manager User's Guide*, available from the Install Center at **`http://support.sas.com/documentation/installcenter/index.html`**.

3 Previously published DS2 code must be modified and republished. Modify the PROC DS2 NOLIBS statement by referring to the example below. The required changes include removing SAS_ENCRYPT=YES and adding NOPROMPT and DRIVER=DS2. Republish the modified DS2 code to the new DataFlux Federation Server. Here is an example of the PROC DS2 NOLIBS statement.

```
proc ds2 nolibs noprompt="driver=remts;server=<federation server machine>;
port=<port>;protocol=bridge;uid=<federation server admin>;
pwd= <federation server admin password>';
conopts=(driver=ds2;CONOPTS=(DSN=BASE_DSN))";
```

Note: If you are upgrading a SAS Decision Services 5.5 deployment that was migrated from SAS Real-Time Decision Manager 5.4, a copy of the SAS Real-Time Decision Manager 5.4 migrated code exists. It was saved in the configuration directory value that was referenced by the ds2.migrate.dir property in the util.properties file that was used as input to the migration utility. Edit the DS2 code that was generated and remove the SAS_ENCRYPT=YES option.

Migrate Single DATA Step SAS Code

The SAS code for SAS activities were DATA step fragments in SAS Real-Time Decision Manager 5.4 and earlier. The code must be migrated to use DS2 package methods in SAS Decision Services 5.5. You must manually execute the migrate script to migrate the SAS code for SAS activities. To migrate SAS activity code, issue the following from the command line:

Note: This is a single command.

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
  5.5\Migrate -k
Enter activity filename here
```

```
C:\SAS\OneMachineRecommended\Lev1\Applications\
  SASDecisionServicesServerConfig5.5\Utilities/util.properties
```

SAS Decision Services Utilities

Various utilities are installed to assist with the migration of system resources and SAS activities. The SAS Decision Services utilities are located in *<SASHome>/* **`SASDecisionServicesServerConfiguration/5.5/`**.

The `ArtifactMigration` utility is provided to migrate the SAS connection system resources that were created in SAS Real-Time Decision Manager 5.4. In previous releases, these resources were used to provide access information about WebSphere MQ. In SAS Decision Services 5.5, these system resources have been removed and are replaced by JDBC Connection system resources. Here is the syntax for the `ArtifactMigration` utility:

```
ArtifactMigration -host <host> -port <port> -user <user> -password
<password> -source <source repositrory> [-target <target repository>]
[-delete] [-create <DEV | TEST | PROD>
```

Option	Description
-host	The name of the SAS Metadata Repository host machine.
-port	The port of the SAS Metadata Repository host machine.
-user	The user name that is used to log on to the SAS Metadata Repository host machine.
-password	The password that is used to log on to the SAS Metadata Repository host machine.
-source	The name of the source repository.
-target	The name of the target repository. If not specified, the source is updated in place.
-delete	If present, the source repository is deleted upon migration completion.

-create <*create*>	If present, the target repository is created prior to migration. Use DEV, TEST, or PROD to indicate the type of the target repository.
-debug	Print debugging information.
-domain <*domain*>	User authentication domain.
-log <*log-file*>	Log file or directory.
-nolog	Disable log file.
-profile <profile>	Metadata server connection profile. Can be used in place of the -host, -port, -user, and -password options.
-version <version>	Used to indicate the root path for the source repository. Default value is 5.5, which should be used if the repository has been imported manually into the 5.5 environment. Valid values are 5.3, 5.4, and 5.5

The `Migrate` utility is provided to migrate SAS Decision Services SAS activities that were created in SAS Real-Time Decision Manager 5.4. In previous releases, these activities were DATA step fragments. In SAS Decision Services 5.5, the SAS activities are written as DS2 package methods. Here is the syntax for the `Migrate` utility.

```
Migrate [-k] [-f] [Activity File OR SASCode Directory]
      Property_Filename
```

Option	**Description**
-k	Keep the temporary files that are created as part of the migration process.
-f	Migrate files based on the sc_programs.sas file.
Activity File	The full path to the single DATA step SAS file to be migrated.

SASCode Directory	The full path to the SASCode directory containing .sas.exe.location, ds2.package.library, and ds2.migrate.dir. This file is usually created during configuration and can be found at <*SASHome*>\SASDecisionServicesServerConfiguration\5.5\util.properties.

The `UpdateResource` utility is provided to easily change information that is stored in the SAS Decision Services Web Service Connection and JDBC Connection type system resources. Here is the syntax for the `UpdateResource` utility.

```
UpdateResource [-?] [-authdomain <current_value new_value>]
 [-authpassword <current_value new_value>]
 [-authuser <current_value new_value>] [-conopts <current_value new_value>]
 [-debug] [-domain <domain>] [-driverclassname <current_value new_value>]
 [-endpointaddress <current_value new_value>] [-host <hostname>]
 [-jdbcpassword <current_value new_value>]
 [-jdbcuser <current_value new_value>] [-log <log-file>] [-nolog]
 [-password <password>] [-port <port>] [-profile <profile>]
 [-proxyhost <current_value new_value>]
 [-proxypassword <current_value new_value>]
 [-proxyport <current_value new_value>]
 [-proxyuser <current_value new_value>]
 [-repository <repository>] [-resource <resource>]
 [-serverurl <current_value new_value>] [-user <userid>]
 [-wsdluri <current_value new_value>]
```

Option	**Description**
-?,-help	Prints help information.
-debug	Prints debugging information.
-log <*log-file*>	Log file or directory.
-nolog	Disable log file.
-repository <*repository*>	The repository name.
-resource <*resource*>	The resource type.

Note: The metadata server connection options are required if -profile is not set or if the profile does not contain connection credentials.

Metadata Server Connection Option	Description
-port <*port*>	Metadata server port.
-domain <*domain*>	User authentication domain.
-host <*hostname*>	Metadata server host.
-password <*password*>	User's login password.
-user <*userid*>	User's login identity.
-profile	Can be used in place of the -host, -port, -user, and -password options.

Web Service Connection Resource Authentication Option	Description
-authdomain <*current_value new_value*>	The domain value.
-authpassword <*current_value new_value*>	The password value.
-authuser <*current_value new_value*>	The user value.
-proxyhost <*current_value new_value*>	The proxy host value.
-proxypassword <*current_value new_value*>	The proxy password value.
-proxyport <*current_value new_value*>	The proxy port value.
-proxyuser <*current_value new_value*>	The proxy user value.
-wsdluri <*current_value new_value*>	The WSDL URI value.

JDBC Option	Description
-driverclassname <*current_value new_value*>	The driver class name value.
-endpointaddress <*current_value new_value*>	The endpoint address value.

JDBC Connection Resource Option	Description
-jdbcpassword <*current_value new_value*>	The password value.

-jdbcuser <*current_value new_value*>	The user value.
-serverurl <*current_value new_value*>	The server URL value.

ConOpts Option	Description
-conopts <*current_value new_value*>	The ConOpts value.

Once you have run the UpdateResource utility, the following items have been migrated:

1 All message queue polling server metadata is removed.

2 All Queue Manager metadata is removed.

3 The repositories SASDSDesignRepository and SASDSEngineRepository and their contents are migrated automatically using the migration utility.

4 The SAS Real-Time Decision Server Config software component is migrated to the metadata folder **`/System/Application/SAS Decision Services/Decision Services 5.5`** and renamed to Decision Services Server Config 5.5.

Migrate Multiple DATA Step Code

Multiple DATA step code must be migrated manually. Skip these steps if your company, or your on-site SAS personnel, has not created new custom multiple DATA step activities. Otherwise, perform the following steps to migrate your multiple DATA step code to a stored process:

1 Convert the SAS code to a stored process that uses input and output parameters as macro variables. In most cases, you remove the %macro and %mend statements from your multiple DATA step SAS code. For code that uses tables and arrays, references to %sc_encode_decode must be changed to %scencode.

2 Load the stored process into the metadata. If WIP is running, it is available as a SAS BI Web service.

3 Modify consuming flows to replace all references of the multiple DATA step SAS activity with the SAS BI Web service activity.

4 Connect the input and output parameters of the flow to the BI Web services using XPath expressions. If you need help with XPath expressions, or with any of the other steps, contact your on-site SAS support personnel, or SAS Technical Support.

5 Open SAS Management Console. Select the metadata profile of a user who has permissions on the design and production repositories.

6 From the **Folder** tab, navigate to **System** ▶ **Applications** ▶ **SAS Decision Services** ▶ **SAS Decision Services 5.5** ▶ **SASDSEngineRepository**.

7 Right-click `SASDSEngineRepository`, and click **New System Resource**.

8 From the drop-down menu, select **Web Service**.

9 Create a Web service resource that points to the URL of the BI Web service (for example, `http://host:port/SASBIWS/services/Shared Data/`*`MyStoredprocesses/myStoredProcess`*).

10 Click **OK**.

11 Repeat the steps for `SASDSDesignRepository`.

12 In both the design and engine repositories, create a Web service activity that points to the Web service resource.

See the *SAS 9.3 BI Web Services Developer's Guide*, available at `http://support.sas.com/documentation/cdl/en/wbsvcdg/62759/PDF/default/wbsvcdg.pdf`, for more information.

Change Host Migration

Federation, Database, and SAS Servers

DataFlux Federation Server and Database Server Manual Migration

Because SAS Decision Services can be configured for access to any schema or table in your database, automatic database server content migration is not possible. DataFlux Federation Server and authentication server content migration is a manual process.

If the database server was migrated to a different host, then load the DS2 packages into the new database server.

If the DataFlux Federation Server and authentication server are migrated to a different host, then you must manually reconfigure the DSNs, data services, users, groups, and base schema and catalog.

JDBC Connection Resources

If the database server host or the DataFlux Federation Server were migrated, then change the server URL for each JDBC Connection resource. You can execute the following commands that perform this task for each JDBC Connection resource. To change the server host name, perform the following steps:

1 Issue the following from the command line:

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
   5.5\UpdateResource
-host Host location -port Port number -user Enter username -password
Enter password -repository SASDSEngineRepository -resource
JDBCConnectionResource -serverurl current_value new_value
```

2 Issue the following from the command line:

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
   5.5\UpdateResource
-host Host location -port Port number -user Enter username -password
```

```
Enter password -repository SASDSDesignRepository -resource
JDBCConnectionResource -serverurl current_value new_value
```

If the database server host or the DataFlux Federation Server were migrated, then you must change the connection options for the DataFlux Federation Server JDBC Connection resources. You can execute commands that perform this task for each JDBC Connection resource. To change the connection options, perform the following steps:

1 Issue the following from the command line:

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
   5.5\UpdateResource
-host Host location -port Port number -user Enter username -password
Enter password -repository SASDSEngineRepository -resource
JDBCConnectionResource -conopts current_value new_value
```

2 Issue the following from the command line:

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
   5.5\UpdateResource
-host Host location -port Port number -user Enter username -password
Enter password -repository SASDSDesignRepository -resource
JDBCConnectionResource -conopts current_value new_value
```

Web Service Resources

If the WIP server was migrated, then change the WSDL URL for each Web service resource. You can create a batch file that performs this task for each Web service resource. To change the endpoint address, perform the following steps:

1 Issue the following from the command line:

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
   5.5\UpdateResource
-host Host location -port Port number -user Enter username -password
Enter password -repository SASDSEngineRepository -resource
WSConnectionResource -wsdluri current_value new_value
```

2 Issue the following from the command line:

```
C:\Program Files\SASHome\SASDecisionServicesServerConfiguration\
   5.5\UpdateResource
-host Host location -port Port number -user Enter username -password
```

```
Enter password -repository SASDSDesignRepository -resource
WSConnectionResource -wsdluri current_value new_value
```

3 Redeploy the BI Web services for your SAS Decision Services stored processes.

9

Best Practices

SAS Server Options

Some options can be edited to improve the performance of the DataFlux Federation Server. It is recommended that the logging be left on while setting up the server, and then turned off for production. When turned on, logging significantly reduces overall system performance. The logging levels that affect your DS2 activities are specified by the log4sas configuration file installed with your DataFlux Federation Server.

By default, the level is set to **`Info`**. For production, the application logger should be changed to **`Error`**.

```
<!-- Application message logger -->
<logger name="App">
       <level value="Info"/>
 </logger>
```

After you disable logging, you must restart the DataFlux Federation Server.

JDBC Performance Tuning

Tuning Controls

Two JDBC Connection resources are configured for four separate purposes:

- To allow decision service activities to execute DS2 package methods, which are hosted by DataFlux Federation Server.
- To allow scoring activities to execute DS2 scoring packages, which are hosted by DataFlux Federation Server.
- To enable General I/O activities to read from and write to relational databases such as Oracle, DB2, Teradata, and Microsoft SQL Server.
- To enable General I/O activities, that are configured to access SAS data sets, to read SAS data sets through DataFlux Federation Server Base Driver.

SAS Decision Services uses Apache Commons Database Connection Pooling (DBCP) to affect efficient caching and management of JDBC Connections, parameterized prepared statements, and parameterized callable statements. For more information about Apache Commons DBCP, see `http://commons.apache.org/dbcp/index.html`.

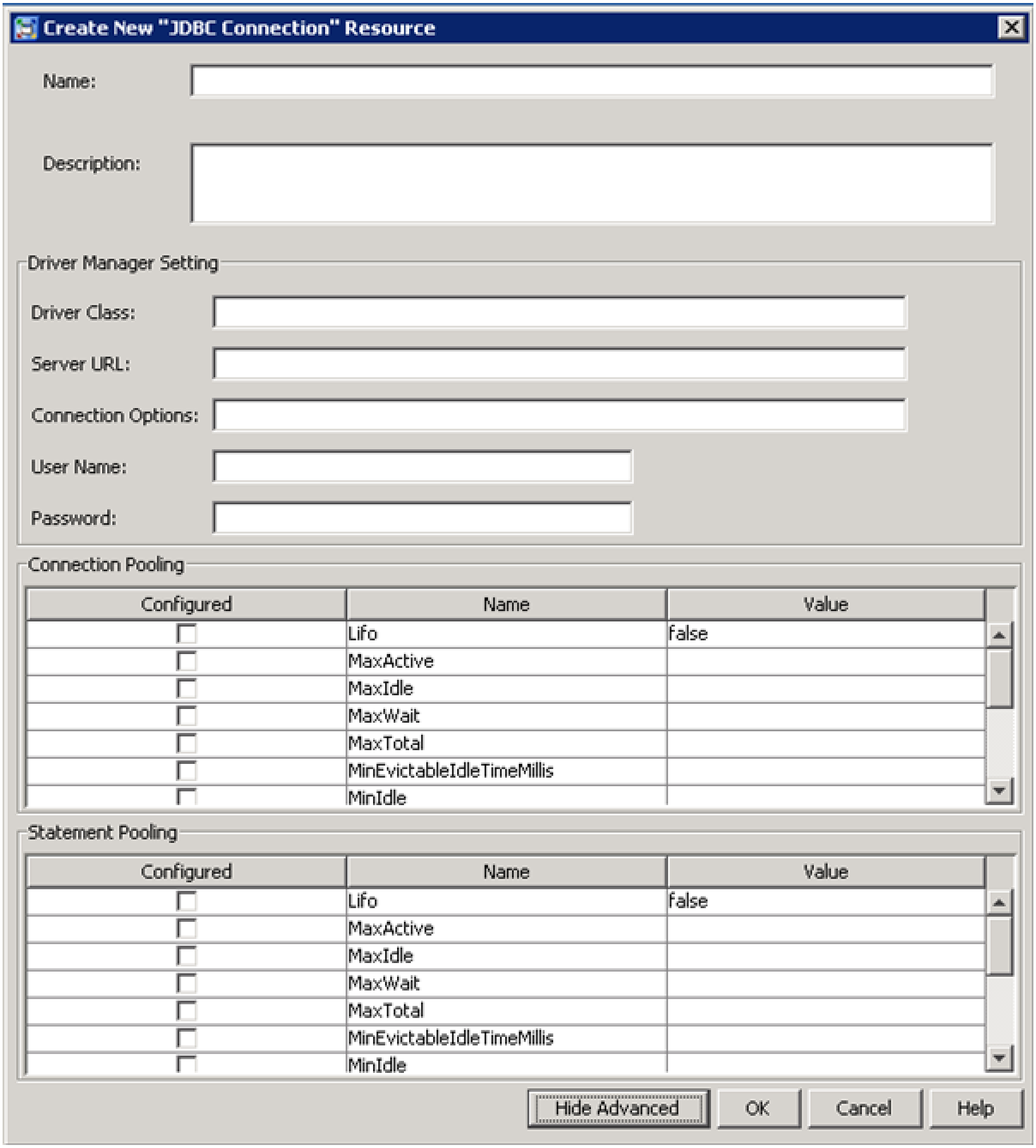
DBCP pool tuning values are stored in JDBC Connection resources. To access the pool tuning controls, select the **Folders** tab, and follow these steps:

1. On the Folders tab, expand **System** ▸ **Applications** ▸ **SAS Decision Services** ▸ **Decision Services 5.5**.

2. Right-click the JDBC Connection resource that you want to configure, and select **Edit System Resource**.

At a minimum, two JDBC Connection resources are needed to satisfy the different pool settings required for optimum DataFlux Federation Server performance versus the settings required for optimum relational database performance. Additional JDBC Connection resources can be added to enable access to additional database management systems, or to assign specific activities to specific DataFlux Federation Servers.

Display 9.1 Create a JDBC System Resource - Advanced



Create New "JDBC Connection" Resource

Name:

Description:

Driver Manager Setting

Driver Class:

Server URL:

Connection Options:

User Name:

Password:

Connection Pooling

Configured	Name	Value
□	Lifo	false
□	MaxActive	
□	MaxIdle	
□	MaxWait	
□	MaxTotal	
□	MinEvictableIdleTimeMillis	
□	MinIdle	

Statement Pooling

Configured	Name	Value
□	Lifo	false
□	MaxActive	
□	MaxIdle	
□	MaxWait	
□	MaxTotal	
□	MinEvictableIdleTimeMillis	
□	MinIdle	

Each attribute can be disabled or enabled by selecting the check box in the Configured column. You can also enable or disable all attributes by selecting **Enable All**. If all of the attributes are disabled in either Connection Pooling or Statement Pooling, the XML element is not be created in the JDBC resource. If the pooling control is saved with the JDBC resource, you see the advanced dialog box when you edit this system resource the next time. You can click **Reset to Default** to return to the basic dialog box.

After you click **OK**, the new JDBC Connection system resource appears in the repository.

The terms and definitions that follow are also listed in the Help for this dialog box.

Lifo

determines whether the pool returns idle objects in last-in-first-out order. The default setting for this parameter is `true`.

MaxActive

controls the maximum number of objects that can be allocated by the pool (checked out to clients or idle awaiting check-out) at a given time. When the value is non-positive, there is no limit to the number of objects that can be managed by the pool at one time. When MaxActive is reached, the pool is said to be exhausted. The default setting for this parameter is 8.

MaxIdle

controls the maximum number of objects that can sit idle in the pool at any time. When the value is negative, there is no limit to the number of objects that can be idle at one time. The default setting for this parameter is 8.

MaxWait

the maximum amount of time, in milliseconds, to wait for an idle object when the pool is exhausted. The default setting for this parameter is -1, which means the wait can continue indefinitely.

MaxTotal

sets a global limit on the number of objects that can be in circulation (active or idle) within the combined set of pools. When the value is non-positive, there is no limit to the total number of objects in circulation. When maxTotal is exceeded, all keyed pools are exhausted. When maxTotal is set to a positive value, and borrowObject is invoked when at the limit with no idle instances available, an attempt is made to create room by clearing the oldest 15% of the elements from the keyed pools. The default setting for this parameter is -1 (no limit).

MinEvictableIdleTimeMillis

specifies the minimum amount of time that an object can sit idle in the pool before it is eligible for eviction because of idle time. When the value is non-positive, no object is dropped from the pool because of idle time alone. This setting has no effect unless TimeBetweenEvictionRunsMillis is greater than 0. The default setting for this parameter is 30 minutes.

MinIdle

sets a target value for the minimum number of idle objects (per key) that should always be available. If this parameter is set to a positive number and timeBetweenEvictionRunsMillis is greater than 0, each time the idle object eviction thread runs, it tries to create enough idle instances so that the specified number of idle instances will be available under each key. This parameter is also used by preparePool, if `true` is provided as that method's populateImmediately parameter. The default setting for this parameter is 0.

NumTestsPerEvictionRun

determines the number of objects to be examined in each run of the idle object evictor. This setting has no effect unless TimeBetweenEvictionRunsMillis is greater than 0. The default setting for this parameter is 3.

TestOnBorrow

whether to validate objects before they are returned by the borrowObject() method.

TestOnReturn

whether to validate objects after they are returned to the returnObject(java.lang.Object) method.

TestWhileIdle

whether to validate objects in the idle object eviction thread, if any.

TimeBetweenEvictionRunsMillis

the amount of time (in milliseconds) to sleep between examining idle objects for eviction.

WhenExhaustedAction

specifies the behavior of the borrowObject() method when the pool is exhausted.

SoftMinEvictableIdleTimeMillis

the minimum number of milliseconds an object can sit idle in the pool before it is eligible for eviction. There is an extra condition that at least MinIdle number of objects remain in the pool.

Tuning Considerations for DataFlux Federation Servers

A DataFlux Federation Server maintains a thread of execution per open JDBC Connection. Therefore, the size of the connection pool has a direct effect on the number of threads that are available to service requests for activity method execution. For best performance, you want the DataFlux Federation Server to maintain an optimum number of ready-to-run threads. For this reason, maxIdle and maxActive should be set to the same value, so that idle connections are not closed, and you want this value to equal the optimum number of threads. Because of the wide variation in server capabilities, you might need to experiment to find this optimum number. A good starting point is to set maxIdle and maxActive equal to the number of cores in the DataFlux Federation Server. Adjust this number up and down while measuring CPU use, latency, and throughput in order to achieve an optimal setting.

The size of the statement pool should be large enough to contain an entry for every activity method deployed to the system. A statement pool is allocated per connection, so do not multiply the number of statements by the number of connections. Instead, simply use the number of statements as the maxActive value. If memory is at a premium, the maxIdle value can be adjusted down to reclaim space taken up by methods that are only rarely called.

Tuning Considerations for Database Management Systems

Database performance tuning is a highly complex and specialized topic beyond the scope of this document. It depends on many factors, including network bandwidth, cache size, data transfer rates, disk array configuration, application characteristics, and more. See your database management system vendor for assistance.

Keeping Connections Alive

A database connection is essentially a socket connection. Operating systems, database hosts, and firewalls drop idle TCP connections after a period of time. The following settings, which are not default, run the evictor every 30 minutes and evict any idle connections older than 30 minutes. This prevents connection failures because of invalid TCP connections, in most cases. The connection pools are reset when the DataFlux Federation Server connections are refreshed.

```
<TimeBetweenEvictionRunsMillis>1000 * 60 * 30</TimeBetweenEvictionRunsMillis>

<NumTestsPerEvictionRun>3</NumTestsPerEvictionRun>

<MinEvictableIdleTimeMillis>1000 * 60 * 30</MinEvictableIdleTimeMillis>
```

There is a one-to-one relationship between the number of request processing threads that are allocated within DataFlux Federation Server and the size of the DataFlux Federation Server JDBC Connection pool. Therefore, pool size affects performance.

See your database vendor for information about performance tuning JDBC Connection and statement pools.

Use of SAS Data Sets

SAS data sets can be used by SAS Decision Services to hold data, as well as DS2 activity code and score code. To access data held in SAS data sets, you must configure a DSN in the DataFlux Federation Server that uses the Base driver, and then use this DSN in the resource definition.

The folder that contains the data sets must be accessible to the DataFlux Federation Server. If multiple DataFlux Federation Servers are in use to implement failover or load balancing, then the data sets must be held in a shared folder that is accessible to all of the DataFlux Federation Servers. This requires the DataFlux Federation Server to have Read access to this shared folder.

Because SAS data sets do not support concurrent access for writing to them, they are commonly used as read-only data through the General I/O activity to preserve performance.

It is possible to write to SAS data sets through the DataFlux Federation Server by disabling concurrent access. This is done by limiting the topology to include a single DataFlux Federation Server and a single engine node, and then setting the size of the connection pool for the resource to 1. Reducing the number of connections reduces the number of threads for this connection and affects throughput for this resource. For applications that must write data, it is strongly recommended to use a relational database that supports concurrent writes. This includes usages like audit logging and user logging.

SAS data sets can also be used to hold DS2 activity code and score code. These are read by the DataFlux Federation Server when a new connection is created. As mentioned earlier, for configurations using multiple DataFlux Federation Servers, the data sets must be held in a shared folder accessible by all of the DataFlux Federation Servers, and requires the DataFlux Federation Server to have Read access to this shared folder.

Score code data sets are created and updated by the score code publishing stored procedure. If the SAS stored process server that executes the score code publishing code is not co-located with the DataFlux Federation Server, the scoring data sets must be held in a shared folder and the score code publishing stored procedure requires read/write access to this folder.

10

Custom Configuration

Standard System Resources

When your system was initially installed and configured, several system resources were created by default.

SAS_Activity_Resource

Configured to reference a DataFlux Federation Server for the purpose of executing SAS activities. SAS activity code is contained in DS2 packages. By default, these DS2 packages are stored in SAS data sets. It is possible to reconfigure the system to store activity DS2 packages in a third-party database, if you so choose. Contact your SAS on-site support personnel for assistance.

$Score_JDBCConnectionResource

Configured to reference a DataFlux Federation Server for the purpose of executing scoring code. Each scoring function corresponds to a predictive model. The score code is contained in a DS2 package. Each scoring DS2 package has the same name as the predictive model that it implements and contains only one function, called score(). By default, these DS2 packages are stored in SAS data sets. It is possible to reconfigure the system to store scoring DS2 packages in a third-party database, if you so choose. Contact your SAS on-site support personnel for assistance.

GeneralIO_Activity_Resource_FS

Configured to reference a DataFlux Federation Server for the purpose of reading SAS data sets. Reading SAS data sets is optional and no activity is configured to use this system resource by default. SAS Decision Services can access SAS data sets only through a DataFlux Federation Server. See "Configure Access to SAS Data Sets" on page 151, for more information.

Note: By default, SAS_Activity_Resource, $Score_JDBCConnectionResource, and GeneralIO_Activity_Resource_FS are configured identically. Unless you have modified the system configuration to allocate different hardware resources for SAS activity execution, scoring, or for SAS data sets, you should maintain these three system resources in parallel. If you add a DataFlux Federation Server to one of these resources, you should also add the same DataFlux Federation Server to the other two.

GeneralIO_Activity_Resource

Configured to reference a third-party database management system for the purpose of storing and accessing data. By default, this system resource is configured to reference the database that was chosen during configuration. Also, by default, the General I/O activity is configured to use this system resource.

$Audit_Log_JDBCConnectionResource

Configured to reference a third-party database management system for the purpose of logging auditing information. Decision service activation, deactivation, and changes that were made to the values of global variables are recorded by the audit log. By default, this system resource is configured to reference the third-party database that was chosen during configuration.

$User_Log_JDBCConnectionResource

Configured to reference a third-party database management system for the purpose of logging performance data. See "Data Collection for Performance Analysis" on page 43, for details. By default, this system resource is configured to reference the third-party database that was chosen during configuration.

Configuring Additional Databases

Overview

During installation, standard SAS Decision Services deployments are configured for access to one third-party database management system and for access to SAS data sets. (Optional) Access to additional third-party database management systems can be configured.

Note: These instructions assume the additional database is to be used for data storage and access only, and not for use by DataFlux Federation Server to read DS2 packages.

Note: It is possible to access databases through a DataFlux Federation Server. However, doing so results in degraded performance. Instead, configure SAS Decision Services to use the native JDBC driver provided by your database vendor.

Your installation might include one or more development, test, or production environments. Repeat the procedures described in this section for each environment that you want to add the additional database to.

Database Installation Checklist

Complete the following checklist. This information is used in “Install Database Client and Server Software” on page 147.

Table 10.1 *SAS Decision Services Checklist — Database Information*

	Defaults	
Database Type:		Choose One: __DB2 9.5 or later __MS SQL 2008 or later __Oracle 10g or later __Teradata 12.0 or later __SAS
Database Host:		
Database Port:		
Database Name:		
Database User ID:		
Database Password:		

Install Database Client and Server Software

SAS Decision Services uses Apache DBCP for its JDBC Connection and statement pooling implementation. Therefore, the connection pooling services of the application server are not used.

The goal of installing the database client software is to make the native JDBC driver class for your database accessible to SAS Decision Services.

For JBoss, you must copy the JDBC driver JAR files to the application server's lib directory.

For WebSphere, you must configure a JDBC provider for your database. This is accomplished within the WebSphere administrator's console, where you provide the path to the JDBC driver JAR files.

For WebLogic, you must copy the JDBC driver JAR files to a specific location and the addition of paths to the start-up scripts.

To install the database client and server software:

1 Install the required database management system server software on a designated database server machine, and then configure your database server. See the installation documentation for the specific database.

2 Follow your application server vendor's instructions to deploy the JDBC driver for your selected database. This procedure makes the database driver available for use by the SAS Decision Services engine.

For information about adding JDBC drivers to WebLogic Application Server, see `"Creating and configuring a JDBC provider and data source using the Java Management Extensions API,"` which is found in the WebSphere Application Server information center, available at `http://publib.boulder.ibm.com/` (the IBM Support Portal).

If, when you are creating a new JDBC provider for WebSphere, your database vendor does not appear in the **Database Type** drop-down list, select **User-Defined** and follow the prompts.

For more information about adding JDBC drivers to JBoss Application Server, see `"Chapter 13. Using other Databases,"` which is found in the jboss.org Community Documentation, available at **`http://docs.jboss.org/`**.

For more information about adding a JDBC provider to WebSphere Application Server, see `"Configuring a JDBC provider using the administrative console,"` which is found in the WebSphere Application Server information center, available at **`http://publib.boulder.ibm.com/`** (the IBM Support Portal).

Define Database Users

Defines a database user ID with create, alter, and delete table permissions that is used to access data from the database and is also used as the Java Access user ID.

Define a JDBC Connection System Resource

For information about defining a JDBC Connection system resource, see "Specify a New System Resource as a JDBC Connection" on page 55.

(Optional) Create Schema Aliases

For more about creating schema aliases, see "(Optional) Define a Schema Alias" on page 61.

(Optional) Specify Tables to Cache in Memory

For more information about specifying tables to cache in memory, see "(Optional) Specify Tables to Cache in Memory" on page 62.

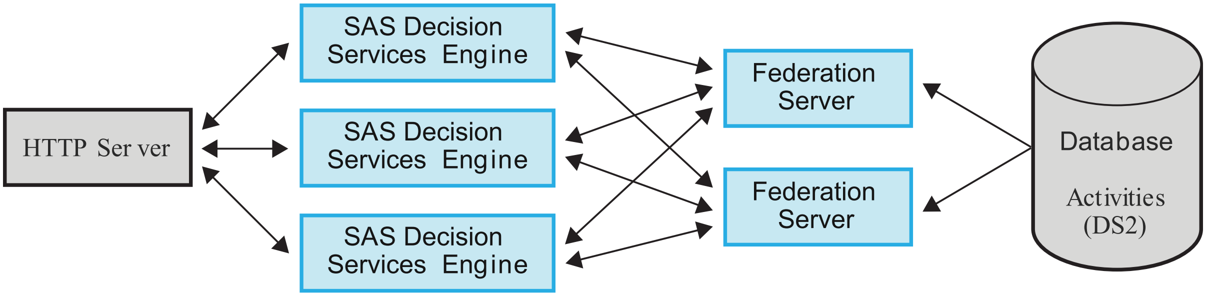
Configuring Additional DataFlux Federation Servers to Form a Server Cluster

Overview

The standard SAS Decision Services installation and deployment process configures a single engine middle tier and a single DataFlux Federation Server. For production deployments, more than one middle-tier engine and more than one DataFlux Federation Server are typically configured to meet system performance and availability requirements. See Best Practices on page 133 for information about how to determine the number of servers to allocate to each tier. A minimum of two DataFlux Federation Servers are required to support hardware failover.

The SAS Decision Services engine attempts to load balance every DataFlux Federation Server for which there is a corresponding URL in the JDBC Connection system resource that is used for executing activities. Therefore, for proper operation, every such DataFlux Federation Server must have access to the same set of DS2 packages. This requires that every such DataFlux Federation Server be configured identically, with the same logins, database users, and DSNs. If your DS2 packages are stored in SAS data sets, those data sets must be located in a shared directory that is accessible from all such DataFlux Federation Servers.

Figure 10.1 *Load Balancing*


HTTP Ser ver
SAS Decision Services Engine
SAS Decision Services Engine
SAS Decision Services Engine
Federation Server
Federation Server
Database
Activities (DS2)

Install and Configure a New DataFlux Federation Server

See the *SAS Decision Services 5.5 Configuration Guide* for information about installing and configuring a DataFlux Federation Server. Be sure to configure the new DataFlux Federation Server identically to the one that is currently used to execute activities. It must have access to the same set of DS2 packages, and it must have the same logins, database users, and DSNs defined.

Edit JDBC Connection System Resources

Use the following procedure to edit your JDBC Connection System Resources to recognize the new DataFlux Federation Server that you configured earlier. If you are modifying a standard configuration, repeat this procedure for each of the system resources:

- SAS_Activity_Resource
- $Score_JDBCConnectionResource
- GeneralIO_Activity_Resource_FS

Open SAS Management Console, select the **Folders** tab, and follow these steps:

1. Expand **System** ▶ **Applications** ▶ **SAS Decision Services** ▶ **Decision Services 5.5**.
2. Select the repository folder.
3. Right-click the desired system resource, and select **Edit System Resource** from the drop-down menu.
4. Modify the **Server URL** field only, by adding a space followed by the full URL, including protocol, of your new DataFlux Federation Server.
5. Click OK to save your changes.

Clustering Best Practices

- If DS2 packages are stored in SAS data sets (the default configuration), they must be located on a shared drive that is accessible by all DataFlux Federation Servers in the cluster. Otherwise, run-time errors occur. Similarly, if the DataFlux Federation Server cluster is used to read data from SAS data sets, all servers must have access to the data sets on a shared drive.
- Multiple DataFlux Federation Servers, which are listed in the Server URL field of a system resource, are uniformly load balanced. Therefore, it is important to deploy each DataFlux Federation Server in a given cluster on the same class of hardware. If this practice is not complied with, the more powerful servers in the cluster is underused while the less powerful servers are overburdened.
- If DS2 packages are stored in a third-party database (a custom configuration), all DataFlux Federation Servers in the cluster must have access to the database and to the same DS2 packages.
- It is possible to configure multiple clusters of DataFlux Federation Servers, in order to allocate hardware resources to specific processing tasks. For assistance with advanced configurations such as this, contact your on-site SAS support personnel for assistance.

Configure Access to SAS Data Sets

To access SAS data sets from SAS Decision Services, create an activity that references GeneralIO_Activity_Resource_FS and include that activity in a decision service.

Except under special circumstances, SAS data sets should be used only for reading data. It is possible to create decision services that write SAS data sets, but in general this practice should be avoided. SAS Decision Services is multi-threaded, and capable of executing multiple Read or Write operations concurrently. SAS data sets have file-level locking, so attempts to write data sets from multiple concurrent threads results in deadlocks and possibly loss of data. If you must write to SAS data sets, then set the connection pool values of both MaxActive and MaxIdle to 1 in the GeneralIO_Activity_Resource_FS system resource. This causes I/O operations to be serialized but to perform slowly. If you must write data, the use of one of the supported database management systems is highly recommended. Concurrent reads of SAS data sets do not cause locking issues.

To configure access to SAS data sets:

1. Using your client application, create a new activity of type GeneralIO.

2. Give the new activity a unique name.

3. (optional) Enter the purpose of the activity in the **Description** field. (For example, “General I/O for reading SAS data sets.”)

4. Enter GeneralIO_Activity_Resource_FS in the systemResourceName field.

Your new activity type is now be visible within your client application’s decision service design user interface.

Appendix 1
Database Requirements

The following table shows the DataFlux Federation Server requirements:

Table A1.1 *Supported Database Clients*

Database Client	Native Driver	Specific Version
DB2 V8.2 Fixpack 9 client and later	DB2	DB2 9.7 client
Oracle 10g client and later	Oracle Wire Protocol	Oracle 11.2 client and administrator and Oracle Enterprise Manager
TTU 12 and later (client and utilities, including TPT)	Teradata	Teradata Client 13.0
Microsoft SQL Server 2000 and later	SQL Server Classic Wire Protocol	SQL Server Native Client 10.0

Here are the SAS 9.3 requirements:

- DB2 Universal Database, Version 8.1 FixPak 18 or later (64-bit libraries)
- Microsoft SQL Server requires a 64-bit ODBC driver
- The minimum required Oracle Client release is Oracle, Release 10g (64-bit libraries)
- Teradata Database 12 or later, Teradata CLIv2 client libraries, TTU 12 or later

Appendix 2

Migration from SAS Real-Time Decision Manager 5.3M3 to 5.4

Overview

Enhancements were made between the third maintenance release for SAS Real-Time Decision Manager 5.3 (M3) and SAS Real-Time Decision Manager 5.4 to the Single DATA Step (SDS) and Multiple DATA Step (MDS) servers. The changes remove naming conflicts that can occur in SAS Real-Time Decision Manager 5.3M3 servers. The value returned (output) for DateTime variables has also been changed. In SAS Real-Time Decision Manager 5.3M3, they came in as numeric but were returned as characters, using the DATETIME19.0 format. To eliminate that inconsistency, DateTime values in SAS Real-Time Decision Manager 5.4 were returned as numeric.

SDS Server

Convert SAS Real-Time Decision Manager 5.3M3 SDS SAS Code to SAS Real-Time Decision Manager 5.4

1. Update all references to hash object sc_input_values from using the single key sc_name to using the multiple key of sc_name, sc_row_number, sc_column_name. See "Input" on page 157, for specifics for each data type as well as examples.

2. Update all references to hash object sc_input_values from using the multiple data values of sc_string and sc_number to using the single data value of sc_data_value. See "Input" on page 157, for specifics for each data type as well as examples.

3. Update all references to hash object sc_output_values from using the single key sc_name to using the multiple key of sc_name, sc_row_number, sc_column_name. See "Output" on page 162, for specifics for each data type as well as examples.

4. Update all references to hash object sc_output_values from using the multiple data values of sc_string and sc_number to using the single data value of sc_data_value. See "Output" on page 162, for specifics for each data type as well as examples.

5. Update all output variables of type DateTime to return the numeric DateTime value instead of the character string formatted with datetime19.0.

Hash Objects

Changes have been made to the hash objects used for input and output. In SAS Real-Time Decision Manager 5.3M3 there was a single key variable, sc_name, and two data variables, sc_string and sc_number. To properly handle arrays and tables, two additional keys have been added in SAS Real-Time Decision Manager 5.4, sc_row_number and sc_column_name. The data variables have been simplified to a single value for SAS Real-Time Decision Manager 5.4, sc_data_value.

Example Code A2.1 *Example of Real-Time Decision Manager 5.3M3*

```
declare hash sc_input_values();
 sc_rc = sc_input_values.defineKey('sc_name');
 sc_rc = sc_input_values.defineData('sc_string', 'sc_number');
 sc_rc = sc_input_values.defineDone();
```

Example Code A2.2 *Example of Real-Time Decision Manager 5.4*

```
declare hash sc_input_values();
 sc_rc = sc_input_values.defineKey('sc_name', 'sc_row_number',
'sc_column_name');
 sc_rc = sc_input_values.defineData('sc_data_value');
 sc_rc = sc_input_values.defineDone();
```

Input

Scalars

To convert a scalar variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1 Add the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of ‘’ (empty string).

2 Change the right side of the assignment from sc_number or sc_string to sc_data_value.

The following examples illustrate the differences in scalars between SAS Real-Time Decision Manager 5.3M3 and SAS Real-Time Decision Manager 5.4.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.3 *Numeric Values*

```
sc_input_values.find(key: 'inputInt');
inputInt=sc_number;
```

Example Code A2.4 *Character Values*

```
sc_input_values.find(key: 'inputString');
      inputString=sc_string;
```

SAS Real-Time Decision Manager 5.4

Example Code A2.5 Numeric and Character Values

```
sc_input_values.find(key: 'inputInt', key: ., key: '');
      inputInt=sc_data_value;
      sc_input_values.find(key: 'inputString', key: ., key: '');
      inputString=sc_data_value;
```

Arrays

The possibility for name conflicts existed for array values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in Real-Time Decision Manager 5.4.

To convert an array variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. To retrieve the row count, add the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of ‘num.rows’.
2. To access data, add the sc_row_number key with a value for the index into the array and the sc_column_name key with a value of ‘’ (empty string).
3. Convert the right side of any assignments from using sc_number and sc_string to using sc_data_value.
4. (Optional) Add type checking by accessing the data type for the array using an sc_row_number key with a value of . (missing) and an sc_column_name key with a value of ‘array.type’. Supported types are: Int, Float, String, Boolean, or DateTime.

The examples below assume a string array named myArray with two values.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.6 Number of Rows

```
sc_input_values.find(key: 'myArray');
      row_count=sc_number;
```

Example Code A2.7 Data Access

```
sc_input_values.find(key: 'myArray1');
      value_row_1=sc_string;
      sc_input_values.find(key: 'myArray2');
      value_row_2=sc_string;
```

Example Code A2.8 *Type Metadata*

```
    /* Not available in 5.3M3 */
```

SAS Real-Time Decision Manager 5.4

Example Code A2.9 *Number of Rows*

```
sc_input_values.find(key: 'myArray', key: ,, key: 'num.rows');
      row_count=sc_data_value;
```

Example Code A2.10 *Data Access*

```
sc_input_values.find(key: 'myArray', key: 1, key: '');
      value_row_1=sc_data_value;
      sc_input_values.find(key: 'myArray', key: 2, key: '');
      value_row_2=sc_data_value;
```

Example Code A2.11 *Type Metadata*

```
sc_input_values.find(key: 'myArray', key: ,, key: 'array.type');
      type=sc_data_value;
```

Tables

The possibility for name conflicts existed for table values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in Real-Time Decision Manager 5.4.

To convert a table variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. To retrieve the number of columns, use the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of 'num.columns'.

2. To retrieve a column name, use the sc_row_number key with a value for the column number and the sc_column_name key with a value of '' (empty string).

3. To retrieve the column type, use the sc_row_number key with a value of . (missing) and the sc_column_name key with the name of the column for which to retrieve the type. Supported types are: Int, Float, String, Boolean, or DateTime.

4. To retrieve the number of rows, use the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of 'num.rows'.

5 To access data, add the sc_row_number key with a value of the desired row and the sc_column_name key with a value of the name for the desired column.

6 Convert the right side of any assignments from using sc_number and sc_string to using sc_data_value.

The examples below assume a table named myTable with two rows and two columns, myInt of type Int and myString of type String.

SAS Real-Time Decision Manager 5.3M3

In SAS Real-Time Decision Manager 5.3M3, tables were set up as several arrays. The main table array held the number of columns and the column names. Each column was then represented as its own array.

Example Code A2.12 *Number of Columns*

```
sc_input_values.find(key: 'myTable');
      column_count=sc_number;
```

Example Code A2.13 *Column Names*

```
sc_input_values.find(key: 'myTable1'); /* Name of column 1 */
      column_name1=sc_string;
      sc_input_values.find(key: 'myTable2'); /* Name of column 2 */
      column_name2=sc_string;
```

Example Code A2.14 *Column Types*

```
/* Not available in 5.3M3 */
```

Example Code A2.15 *Number of Rows*

```
sc_input_values.find(key: 'myInt');
      row_count=sc_number;
```

or:

```
   sc_input_values.find(key: 'myString');
   row_count=sc_number;
```

Example Code A2.16 *Data Access*

```
sc_input_values.find(key: 'myInt1');
      myInt_row_1=sc_number;
      sc_input_values.find(key: 'myInt2');
      myInt_row_2=sc_number;
```

```
        sc_input_values.find(key: 'myString1');
        myString_row_1=sc_string;
        sc_input_values.find(key: 'myString2');
        myString_row_2=sc_string;
```

SAS Real-Time Decision Manager 5.4

In SAS Real-Time Decision Manager 5.4, tables are able to take advantage of the additional keys to remove all potential name conflicts.

Example Code A2.17 *Number of Columns*

```
  sc_input_values.find(key: 'myTable', key: ., key:
  'num.columns');
        column_count=sc_data_value;
```

Example Code A2.18 *Column Names*

```
  sc_input_values.find(key: 'myTable', key: 1, key: ''); /* Name
  of column 1 */
        column_name1=sc_data_value;
        sc_input_values.find(key: 'myTable', key: 2, key: ''); /* Name
  of column 2 */
        column_name2=sc_data_value;
```

Example Code A2.19 *Column Types*

```
  sc_input_values.find(key: 'myTable', key: ., key: 'myInt'); /*
  Type for column myInt */
        myInt_type=sc_data_value; /* Value of 'Int' */
        sc_input_values.find(key: 'myTable', key: ., key: 'myString');
  /* Type for column myString */
        myString_type=sc_data_value; /* Value of 'String' */
```

Example Code A2.20 *Number of Rows*

```
  sc_input_values.find(key: 'myTable', key: ., key: 'num.rows');
         row_count=sc_data_value;
```

Example Code A2.21 *Data Access*

```
  sc_input_values.find(key: 'myTable', key: 1, key: 'myInt');
        myInt_row_1=sc_data_value;
        sc_input_values.find(key: 'myTable', key: 2, key: 'myInt');
        myInt_row_2=sc_data_value;
        sc_input_values.find(key: 'myTable', key: 1, key: 'myString');
        myString_row_1=sc_data_value;
        sc_input_values.find(key: 'myTable', key: 2, key: 'myString');
        myString_row_2=sc_data_value;
```

Output

Scalars

In addition to the change to the output hash object, the value that should be returned for a DateTime variable has also changed. In SAS Real-Time Decision Manager 5.3M3, this was inconsistent in that the DateTime input variables were actual numeric SAS DateTime values. However, DateTime output variables were strings with the datetime19.0 format applied to them. This has been fixed in Real-Time Decision Manager 5.4 so that DateTime variables are numeric SAS DateTime values for both input and output.

To convert a scalar variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. Add the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of ‘’ (empty string).
2. Change the left side of the assignment from sc_number or sc_string to sc_data_value.
3. Change the value of DateTime variables from datetim19.0 format to the DateTime numeric value.

The following examples illustrate the differences in scalars between SAS Real-Time Decision Manager 5.3M3 and SAS Real-Time Decision Manager5.4.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.22 *Numeric Values*

```
sc_output_values.add(key: 'outputInt', data: '', data:
outputInt);
```

Example Code A2.23 *Character Values*

```
sc_output_values.add(key: 'outputString', data: outputString,
data: .);
```

Example Code A2.24 *DateTime Values*

```
sc_string=put(outputDateTime, datetime19.0);
      sc_output_values.add(key: 'outputDateTime', data: sc_string,
data: .);
```

SAS Real-Time Decision Manager 5.4

Example Code A2.25 *Numeric Values*

```
sc_data_value=outputInt; /* Automatic Conversion */
      sc_output_values.add(key: 'outputInt', key: ., key: '', data:
sc_data_value);
```

Example Code A2.26 *Character Values*

```
sc_output_values.add(key: 'outputString', key: ., key: '', data:
outputString);
```

Example Code A2.27 *DateTime Values*

```
sc_data_value=outputDateTime; /* Automatic Conversion */
      sc_output_values.add(key: 'outputDateTime', key: ., key: '',
data: sc_data_value);
```

Arrays

The possibility for name conflicts existed for array values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in Real-Time Decision Manager 5.4.

To convert an array variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. Set the row count using an sc_row_number key with a value of . (missing), the sc_column_name key with a value of ‘num.rows’, and an sc_data_value containing the number of rows.

2. Set the data by using an sc_row_number key with a value for the index into the array and the sc_column_name key with a value of ‘’ (empty string) and an sc_data_value containing the value.

3. Convert the left side of any assignments from using sc_number and sc_string to using sc_data_value.

4 Set the data type for the array using an sc_row_number key with a value of . (missing), an sc_column_name key with a value of ‘array.type’, and an sc_data_value of one of: Int, Float, String, Boolean, or DateTime.

The examples below assume a string array named myArray with two values.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.28 *Number of Rows*

```
sc_output_values.add(key: 'myArray', data: '', data: row_count);
```

Example Code A2.29 *Array Type*

```
/* Not set used in 5.3M3 */
```

Example Code A2.30 *Data Values*

```
sc_output_values.add(key: 'myArray1', data: value_row_1, data:
.);
      sc_output_values.add(key: 'myArray2', data: value_row_2, data:
.);
```

SAS Real-Time Decision Manager 5.4

Example Code A2.31 *Number of Rows*

```
sc_data_value=row_count; /* Automatic Conversion */
      sc_output_values.add(key: 'myArray', key: ., key: 'num.rows',
data: sc_data_value);
```

Example Code A2.32 *Array Type*

```
sc_output_values.add(key: 'myArray', key: ., key: 'array.type',
data: 'String');
```

Example Code A2.33 *Set Data Values*

```
sc_output_values.add(key: 'myArray', key: 1, key: '', data:
value_row_1);
      sc_output_values.add(key: 'myArray', key: 1, key: '', data:
value_row_2);
```

Tables

The possibility for name conflicts existed for table values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in SAS Real-Time Decision Manager 5.4. Also, in SAS Real-Time Decision Manager 5.3M3 the type metadata for the table had to be set in the output table object in the middle tier. For SAS Real-Time Decision

Manager 5.4, this is no longer necessary. Instead, the type must be set in the activity code. This allows for greater flexibility in the output table.

To convert a table variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. Set the number of columns using an sc_row_number key with a value of . (missing), the sc_column_name key with a value of 'num.columns', and an sc_data_value containing the number of columns.
2. Set each column name using an sc_row_number key with a value for the column number (starting at 1), the sc_column_name key with a value of '' (empty string), and an sc_data_value containing the name of the column.
3. Set each column type using an sc_row_number key with a value of . (missing), the sc_column_name key with the name of the column for which to set the type, and an sc_data_value containing one of: Int, Float, String, Boolean, or DateTime.
4. Set the number of rows using an sc_row_number key with a value of . (missing), the sc_column_name key with a value of 'num.rows', and an sc_data_value containing the number of rows.
5. Set the data using an sc_row_number key with a value of the desired row, the sc_column_name key with a value of the name for the desired column, and an sc_data_value containing the actual value.
6. Convert the left side of any assignments from using sc_number and sc_string to using sc_data_value.

The examples below assume a table named myTable with two rows and two columns, myInt of type Int and myString of type String.

SAS Real-Time Decision Manager 5.3M3

In Real-Time Decision Manager 5.3M3, tables were set as several arrays. The main table array held the number of columns and the column names. Each column was then represented as its own array.

Example Code A2.34 *Number of Columns*

```
sc_output_values.add(key: 'myTable', data: '', data:
column_count);
```

Example Code A2.35 *Column Names*

```
sc_output_values.add(key: 'myTable1', data: 'myInt', data: .);
      sc_output_values.add(key: 'myTable2', data: 'myString', data:
.);
```

Example Code A2.36 *Column Types*

```
/* Not set in 5.3M3 */
```

Example Code A2.37 *Number of Rows*

```
sc_output_values.add(key: 'myInt', data: '', data: row_count);
     sc_output_values.add(key: 'myString', data: '', data:
row_count);
```

Example Code A2.38 *Data Values*

```
sc_output_values.add(key: 'myInt1', data: '', data:
myInt_row_1);
      sc_output_values.add(key: 'myInt2', data: '', data:
myInt_row_2);
      sc_output_values.add(key: 'myString1', data: myString_row_1,
data: .);
      sc_output_values.add(key: 'myString2', data: myString_row_2,
data: .);
```

SAS Real-Time Decision Manager 5.4

In Real-Time Decision Manager 5.4, tables are able to take advantage of the additional keys to remove all potential name conflicts.

Example Code A2.39 *Number of Columns*

```
sc_data_value=column_count; /* Automatic Conversion */
      sc_output_values.add(key: 'myTable', key: ., key: 'num.columns',
data: sc_data_value);
```

Example Code A2.40 *Column Names*

```
sc_output_values.add(key: 'myTable', key: 1, key: '', data:
'myInt');
      sc_output_values.add(key: 'myTable', key: 2, key: '', data:
'myString');
```

Example Code A2.41 *Column Types*

```
sc_output_values.add(key: 'myTable', key: ., key: 'myInt', data:
'Int');
      sc_output_values.add(key: 'myTable', key: ., key: 'myString',
data: 'String');
```

Example Code A2.42 *Number of Rows*

```
sc_data_value=row_count; /* Automatic Conversion */
      sc_output_values.add(key: 'myTable', key: ., key: 'num.rows',
data: sc_data_value);
```

Example Code A2.43 *Data Values*

```
sc_data_value=myInt_row_1; /* Automatic Conversion */
     sc_output_values.add(key: 'myTable', key: 1, key: 'myInt', data:
sc_data_value);
     sc_data_value=myInt_row_2; /* Automatic Conversion */
     sc_output_values.add(key: 'myTable', key: 2, key: 'myInt', data:
sc_data_value);
     sc_output_values.add(key: 'myTable', key: 1, key: 'myString',
data: myString_row_1);
     sc_output_values.add(key: 'myTable', key: 2, key: 'myString',
data: myString_row_2);
```

MDS Server

Overview

Changes have been made to the macro variables used for arrays and tables. In SAS Real-Time Decision Manager 5.3M3 each value had its own macro variable. This allowed for the possibility of name conflicts. In SAS Real-Time Decision Manager 5.4, array and table macro variables are encoded strings that can be decoded and re-encoded using the provided macro and link statements.

To convert your SAS Real-Time Decision Manager 5.3M3 MDS SAS code to SAS Real-Time Decision Manager 5.4 SDS SAS code:

1 Move all references to array and table variables into a DATA step.

2. Add %sc_encode_decode to the beginning of any data step that will use array or table variables.

3. Decode input arrays or tables prior to use. See Arrays or Tables in "Input" on page 168.

4. Encode output arrays and tables prior to finishing the DATA step. See Arrays or Tables in "Output" on page 173.

Input

Scalars

There are no changes to scalars in SAS Real-Time Decision Manager 5.4.

Arrays

The possibility for name conflicts existed for array values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in SAS Real-Time Decision Manager 5.4.

To convert an array variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. Move the array references into a DATA step.

2. Add %sc_encode_decode as the first line in the DATA step.

3. Set sc_name equal to the name of the array variable.

4. Call link sc_decode_array. The array is now available in the sc_input_values hash object.

5. Retrieve the number of rows by using an sc_row_number key with a value of . (missing) and the sc_column_name key with a value of 'num.rows'.

6. Retrieve the data by using an sc_row_number key with a value for the index into the array and the sc_column_name key with a value of '' (empty string).

7 Convert the right side of any assignments from using a reference to a macro variable to using sc_data_value.

8 (Optional) Add type checking by retrieving the data type for the array using an sc_row_number key with a value of . (missing) and an sc_column_name key with a value of ‘array.type’.

The examples below assume a string array named myArray with two values.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.44 *Number of Rows*

```
row_count=symget('myArray');
```

Example Code A2.45 *Array Type*

```
/* Not available in 5.3M3 */
```

Example Code A2.46 *Data Access*

```
value_row_1=symget('myArray1');
     value_row_2=symget('myArray2');
```

SAS Real-Time Decision Manager 5.4

Example Code A2.47 *Number of Rows*

```
data _null_;
           %sc_encode_decode
           sc_name='myArray';
           link sc_decode_array;

           sc_input_values.find(key: 'myArray', key: ., key:
'num.rows');
           row_count=sc_data_value;
      run;
```

Example Code A2.48 *Array Type*

```
data _null_;
           %sc_encode_decode

           sc_name='myArray';
           link sc_decode_array;

           sc_input_values.find(key: 'myArray', key: ., key:
'array.type');
```

```
              type=sc_data_value; /* Value of 'String' */
       run;
```

Example Code A2.49 *Data Access*

```
data _null_;
          %sc_encode_decode

          sc_name='myArray';
          link sc_decode_array;

          sc_input_values.find(key: 'myArray', key: 1, key: '');
          value_row_1=sc_data_value;
          sc_input_values.find(key: 'myArray', key: 2, key: '');
          value_row_2=sc_data_value;
     run;
```

Tables

The possibility for name conflicts existed for table values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in SAS Real-Time Decision Manager 5.4.

To convert a table variable from SAS Real-Time Decision Manager 5.3M3 to SAS Real-Time Decision Manager 5.4:

1. Move the table reference into a DATA step.
2. Add %sc_encode_decode as the first line in the DATA step.
3. Set sc_name equal to the name of the table variable.
4. Call link sc_decode_table. The table is now available in the sc_input_values hash object.
5. To retrieve the number of columns, use the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of 'num.columns'.
6. To retrieve a column name, use the sc_row_number key with a value for the column number and the sc_column_name key with a value of '' (empty string).

7 To retrieve the column type, use the sc_row_number key with a value of . (missing) and the sc_column_name key with the name of the column for which to retrieve the type. Supported types are: Int, Float, String, Boolean, or DateTime.

8 To retrieve the number of rows, use the sc_row_number key with a value of . (missing) and the sc_column_name key with a value of 'num.rows'.

9 To access data, add the sc_row_number key with a value of the desired row and the sc_column_name key with a value of the name for the desired column.

10 Convert the right side of any assignments from using macro variables to using sc_data_value.

The examples below assume a table named myTable with two rows and two columns, myInt of type Int and myString of type String.

SAS Real-Time Decision Manager 5.3M3

In SAS Real-Time Decision Manager 5.3M3, tables were setup as several arrays. The main table array held the number of columns and the column names. Each column was then represented as its own array.

Example Code A2.50 *Number of Columns*

```
column_count=symget('myTable');
```

Example Code A2.51 *Column Names*

```
column_name1=symget('myTable1');
     column_name2=symget('myTable2');
```

Example Code A2.52 *Column Types*

```
/* Not available in 5.3M3 */
```

Example Code A2.53 *Number of Rows*

```
row_count=symget('myInt');
```

or:

```
  row_count=symget('myString');
```

Example Code A2.54 *Data Access*

```
myInt_row_1=symget('myInt1');
     myInt_row_2=symget('myInt2');
     myString_row_1=symget('myString1');
     myString_row_2=symget('myString2');
```

SAS Real-Time Decision Manager 5.4

Example Code A2.55 *Number of Columns*

```
data _null_;
           %sc_encode_decode

           sc_name='myTable';
           link sc_decode_table;

           sc_input_values.find(key: 'myTable', key: ., key:
'num.columns');
           column_count=sc_data_value;
      run;
```

Example Code A2.56 *Column Names*

```
data _null_;
            %sc_encode_decode

            sc_name='myTable';
            link sc_decode_table;

             sc_input_values.find(key: 'myTable', key: 1, key: ''); /*
Name of column 1 */
             column_name1=sc_data_value;
             sc_input_values.find(key: 'myTable', key: 2, key: ''); /*
Name of column 2 */
             column_name2=sc_data_value;
      run;
```

Example Code A2.57 *Column Types*

```
data _null_;
           %sc_encode_decode

           sc_name='myTable';
           link sc_decode_table;

           sc_input_values.find(key: 'myTable', key: ., key: 'myInt');
/* Type for column myInt */
           myInt_type=sc_data_value; /* Value of Int */
           sc_input_values.find(key: 'myTable', key: ., key:
```

```
'myString'); /* Type for column myString */
           myString_type=sc_data_value; /* Value of String */
      run;
```

Example Code A2.58 *Number of Rows*

```
data _null_;
           %sc_encode_decode

           sc_name='myTable';
           link sc_decode_table;

           sc_input_values.find(key: 'myTable', key: ., key:
'num.rows');
           row_count=sc_data_value;
      run;
```

Example Code A2.59 *Data Access*

```
data _null_;
           %sc_encode_decode

           sc_name='myTable';
           link sc_decode_table;

           sc_input_values.find(key: 'myTable', key: 1, key: 'myInt');
           myInt_row_1=sc_data_value;
           sc_input_values.find(key: 'myTable', key: 2, key: 'myInt');
           myInt_row_2=sc_data_value;
           sc_input_values.find(key: 'myTable', key: 1, key:
'myString');
           myString_row_1=sc_data_value;
           sc_input_values.find(key: 'myTable', key: 2, key:
'myString');
           myString_row_2=sc_data_value;
        run;
```

Output

Scalars

In SAS Real-Time Decision Manager 5.3M3, the DateTime input variables were actual numeric SAS DateTime values. However, DateTime output variables were strings with the datetime19.0 format applied to them. In SAS Real-Time Decision Manager 5.4, DateTime variables are numeric SAS DateTime values for both input and output.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.60 *All Types Excluding DateTime*

```
call symputx('myInt', myInt);
```

Example Code A2.61 *DateTime*

```
 call symputx('myDateTime', put(myDateTime, datetime19.0));
```

SAS Real-Time Decision Manager 5.4

Example Code A2.62 *All Types*

```
call symputx('myInt', myInt);
```

Arrays

The possibility for name conflicts existed for array values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in SAS Real-Time Decision Manager 5.4.

1. Move the array reference into a DATA step.

2. Add %sc_encode_decode as the first line in the DATA step.

3. Set sc_name equal to the name of the array variable.

4. Set the row count using an sc_row_number key with a value of . (missing), the sc_column_name key with a value of ‘num.rows’, and an sc_data_value containing the number of rows.

5. Set the data by using an sc_row_number key with a value for the index into the array, the sc_column_name key with a value of ‘’ (empty string), and an sc_data_value containing the value. Be sure to use the numeric value for DateTime variables. (See “Scalars” on page 173.)

6. Convert the left side of any assignments from setting a macro variable to using sc_data_value.

7. Set the data type for the array using an sc_row_number key with a value of . (missing), an sc_column_name key with a value of ‘array.type’, and an sc_data_value of one of: Int, Float, String, Boolean, or DateTime.

8 Call link sc_encode_array.

The examples below assume a string array named myArray with two values.

SAS Real-Time Decision Manager 5.3M3

Example Code A2.63 *Number of Rows*

```
 call symputx('myArray', row_count);
```

Example Code A2.64 *Array Type*

```
/* Not used in 5.3M3 */
```

Example Code A2.65 *Data Values*

```
call symputx('myArray1', value_row_1);
      call symputx('myArray2', value_row_2);
```

SAS Real-Time Decision Manager 5.4

Example Code A2.66 *Number of Rows*

```
data _null_;
           %sc_encode_decode

           row_count=sc_data_value; /* Automatic Conversion */
           sc_output_values.add(key: 'myArray', key: ., key: 
'num.rows', data: sc_data_value);

           sc_name='myArray';
           link sc_encode_array;
      run;
```

Example Code A2.67 *Array Type*

```
data _null_;
           %sc_encode_decode

           sc_output_values.add(key: 'myArray', key: ., key: 
'array.type', data: 'String');

           sc_name='myArray';
           link sc_encode_array;
      run;
```

Example Code A2.68 *Data Values*

```
data _null_;
           %sc_encode_decode
           sc_output_values.add(key: 'myArray', key: 1, key: '', data:
value_row_1);
           sc_output_values.add(key: 'myArray', key: 2, key: '', data:
value_row_2);

           sc_name='myArray';
           link sc_encode_array;
      run;
```

Tables

The possibility for name conflicts existed for table values in SAS Real-Time Decision Manager 5.3M3. This issue has been resolved in Real-Time Decision Manager 5.4. Also, in Real-Time Decision Manager 5.3M3, the type metadata for the table had to be set in the output table object in the middle tier. For Real-Time Decision Manager 5.4, this is no longer necessary. Instead, the type must be set in the activity code. This allows for greater flexibility in the output table.

1 Move the table reference into a DATA step.

2 Add %sc_encode_decode as the first line in the DATA step.

3 Set sc_name equal to the name of the table variable.

4 Set the number of columns using an sc_row_number key with a value of . (missing), the sc_column_name key with a value of 'num.columns', and an sc_data_value containing the number of columns.

5 Set each column name using an sc_row_number key with a value for the column number (starting at 1), the sc_column_name key with a value of '' (empty string), and an sc_data_value containing the name of the column.

6 Set each column type using an sc_row_number key with a value of . (missing), the sc_column_name key with the name of the column for which to set the type, and an sc_data_value containing one of: Int, Float, String, Boolean, or DateTime.

7 Set the number of rows using an sc_row_number key with a value of . (missing), the sc_column_name key with a value of 'num.rows', and an sc_data_value containing the number of rows.

8 Set the data using an sc_row_number key with a value of the desired row, the sc_column_name key with a value of the name for the desired column, and an sc_data_value containing the actual value.

9 Convert the left side of any assignments from using sc_number and sc_string to using sc_data_value.

10 Call link sc_encode_table.

The following examples assume a table named myTable with two rows and two columns, myInt of type Int and myString of type String.

SAS Real-Time Decision Manager 5.3M3

In SAS Real-Time Decision Manager 5.3M3 tables were set up as several arrays. The main table array held the number of columns and the column names. Each column was then represented as its own array.

Example Code A2.69 *Number of Columns*

```
call symputx('myTable', column_count);
```

Example Code A2.70 *Column Names*

```
call symputx('myTable1', 'myInt');
      call symputx('myTable2', 'myString');
```

Example Code A2.71 *Column Types*

```
/* Not used in 5.3M3 */
```

Example Code A2.72 *Number of Rows*

```
call symputx('myInt', row_count);
      call symputx('myString', row_count);
```

Example Code A2.73 *Data Values*

```
call symputx('myInt1', myInt_row_1);
      call symputx('myInt2', myInt_row_2);
      call symputx('myString1', myString_row_1);
      call symputx('myString2', myString_row_2);
```

SAS Real-Time Decision Manager 5.4

Example Code A2.74 Number of Columns

```
data _null_;
           %sc_encode_decode

           sc_data_value=column_count; /* Automatic Conversion */
           sc_output_values.add(key: 'myTable', key: ., key: 
'num.columns', sc_data_value);

           /* After table has been filled in */
           sc_name='myTable';
           link sc_encode_table;
      run;
```

Example Code A2.75 Column Names

```
data _null_;
           %sc_encode_decode

           sc_output_values.add(key: 'myTable', key: 1, key: '', 
'myInt');
           sc_output_values.add(key: 'myTable', key: 2, key: '', 
'myString');

           /* After table has been filled in */
           sc_name='myTable';
           link sc_encode_table;
      run;
```

Example Code A2.76 Column Types

```
data _null_;
            %sc_encode_decode

            sc_output_values.add(key: 'myTable', key: ., key: 'myInt', 
'Int');
            sc_output_values.add(key: 'myTable', key: ., key: 
'myString', 'String');

            /* After table has been filled in */
            sc_name='myTable';
            link sc_encode_table;
      run;
```

Example Code A2.77 Number of Rows

```
data _null_;
           %sc_encode_decode
           sc_data_value=row_count; /* Automatic Conversion */
           sc_output_values.add(key: 'myTable', key: ., key:
'num.rows', sc_data_value);

           /*After table has been filled in */
           sc_name='myTable';
           link sc_encode_table;
      run;
```

Example Code A2.78 *Data Values*

```
data _null_;
            %sc_encode_decode

            sc_data_value=myInt_row_1; /* Automatic Conversion */
            sc_output_values.add(key: 'myTable', key: 1, key: 'myInt',
sc_data_value);
            sc_data_value=myInt_row_2; /* Automatic Conversion */
            sc_output_values.add(key: 'myTable', key: 2, key: 'myInt',
sc_data_value);
            sc_output_values.add(key: 'myTable', key: 1, key:
'myString', myString_row_1);
            sc_output_values.add(key: 'myTable', key: 2, key:
'myString', myString_row_2);

            /*After table has been filled in */
            sc_name='myTable';
            link sc_encode_table;
        run;
```

Appendix 3
Logs and Troubleshooting

Troubleshooting

Troubleshooting problems with SAS Decision Services requires inspecting the logs generated by the following components:

- SAS Decision Services design server
- SAS Decision Services engine server
- J2EE Application Server
- SAS Management Console
- Open Metadata Server
- Stored Process Server
- DataFlux Federation Server
- DataFlux Authentication Server

Log File Locations

The following table summarizes the location of log files generated by the various SAS Decision Services components.

Table A3.1 Log Locations

Component	Default Location of Logs
Compute Tier	<*SAS Config*> is the location for installing the configuration for SAS. For example, `D:/SAS/Config/Lev1` *yyyy-mm-dd* is the date time of creation of the log file. *nnnn* is a four-digit number to differentiate between logs created by multiple processes that perform the same function. *N.N* is the major and minor version number of SAS Decision Services.
Metadata Server	`<SAS Config>/SASMeta/MetadataServer/Logs/SAS_Meta_MetadataServer_yyyy-mm-dd_nnnn.log`

DataFlux Federation Server	`<install root>\FederationServer\2.1\etc\dfs_log.xml`
	`<install root>\FederationServer\2.1\var\log`

Middle Tier	
Remote Services	`<SAS Config>/Web/Logs/RemoteServices.log`
SAS Decision Services Engine Server	`<SAS Config>/Web/Logs/SASDecisionServicesEngineServerN.N.log`
SAS Decision Services Design Server	`<SAS Config>/Web/Logs/SASDecisionServicesDesignServerN.N.log`
IBM WebSphere Application Server	<*WAS Base*> is the location where WebSphere Application Server is installed, for example, D:/IBM/Websphere/AppServer. <*profile name*> is the name of the server profile. For example, the profile would be called SAS<node name> 01 if <node name> is the name of the node. <*server name*> is the name of the server on which SAS Decision Services Web applications are deployed. By default, both the SAS Decision Services engine and design server are deployed on one called SASServer7, and BI Web services and score publishing code are deployed on one called SASServer1. There are several logs for each server. Details are available in the WebSphere Application Server documentation. For debugging SAS Decision Services errors, the most significant ones are called SystemOut.log and SystemErr.log.

Component	Default Location of Logs
SystemOut	**<*WAS Base*>/profiles/<*profile name*>/logs/<*server name*>/SystemOut.log**
SystemErr	**<*WAS Base*>/profiles/<*profile name*>/logs/<*server name*>/SystemErr.log**
JBoss Application Server	**<*JBoss_Base*>/jboss-as/server/<*server name*>/log/boot.log** **<*JBoss_Base*>/jboss-as/server/<*server name*>/log/server.log** <*JBoss_Base*> is the location where JBoss Application Server is installed. Older server logs are suffixed with a time stamp, for example **<*JBoss_Base*>/jboss-as/server/<*server name*>/log/server.log.yyyy-mm-dd**.
Oracle WebLogic Server	**<*SAS Config*>/Web/SASDomain/servers/<*server name*>/logs/<*server name*>.log** **<*SAS Config*>/Web/SASDomain/servers/<*server name*>/logs/<*server name*>.out** **<*SAS Config*>/Web/SASDomain/servers/<*server name*>/logs/access.log**

Appendix 4
Activate Flows Using BatchActivator

`BatchActivator` is a command-line utility that is used to activate or deactivate decision flows. It can be used either stand-alone or in scripts. The utility requires connecting to the SAS Metadata Repository as part of its operation. The connection information is provided to the utility through command-line parameters. Also, user credentials used by the utility are supplied as command-line parameters, including supplying the credentials in a separate profile file.

When it is scripting, the utility returns a completion code of 0 to indicate success and 8 to indicate an error.

Multiple flows can be activated or deactivated at a time, with the following parameters:

- The input file must contain the list of flow names to be activated or deactivated, one name per line.
- All flows that are specified in the input file must be in the same state. The state can be either inactive (for activation) or active (for deactivation). Otherwise, you receive an error.

The utility updates changes in the repository and notifies the engine about the changes that it makes. If the engine is not reachable by the utility, the flows are deactivated in the repository. However, the utility also supports an option called OFFLINEOK. If the engine is not reachable and you specify the OFFLINEOK option, the changes in the repository are not rolled back.

The following information is logged by the utility:

- The names of all flows that are activated or deactivated.

- Any validation errors that prevent activation .

The following usage statement is printed when the utility is run with the -help option or when the command-line parameters are incorrect:

```
BatchActivator [-?] [-a] [-d] [-debug] [-domain <domain>] [-host <hostname>]
[-l] [-log <log-file>] [-nolog] [-o] [-password <password>] [-port <port>]
[-profile <profile>] [-user <userid>]
```

Options	Descriptions
-?,-help	Prints help information.
-activate	Activates the flows.
-debug	Prints debugging information.
-domain *<domain>*	Provides user authentication domain information.
-host *<hostname>*	Metadata server host. Required if -profile is not set.
-log *<log-file>*	Log file or directory.
-nolog	Disable log file.
-o, --offline	Continue if the engine is off line.
-password *<password>*	User login password. Required if -profile is not set or if the profile does not contain connection credentials.
-port *<port>*	Metadata server port. Required if -profile is not set.
-profile *<profile>*	Metadata server connection profile. Can be used in place of the -host, -port, -user, and -password options.
-user *<userid>*	User login identity. Required if -profile is not set or if the profile does not contain connection credentials.

Glossary

artifact

an element of SAS metadata servers that might contain global variables, activities, events, system resources, or decision flow objects.

campaign

a planned set of one or more communications that are directed at a selected group of customers or potential customers for a commercial goal.

data item

in an information map, an item that represents either data (a table column, an OLAP hierarchy, or an OLAP measure) or a calculation. Data items are used for building queries. Data items are usually customized in order to present the data in a form that is relevant and meaningful to a business user.

data set

See SAS data set

database management system

a software application that enables you to create and manipulate data that is stored in the form of databases. Short form: DBMS.

DBMS

See database management system

federated DSN

a data source name that references multiple data sources. The data sources can be on the same DBMS, or on a different one.

grouping data source name

See federated DSN

grouping DSN

See federated DSN

log

See log file

log file

a file in which information about software processing is recorded as the processing occurs. A log file typically includes error messages and warning messages, but it can also include informational messages and statistics such as the number of records that have been processed or the amount of CPU time that a program required.

macro variable

a variable that is part of the SAS macro programming language. The value of a macro variable is a string that remains constant until you change it. Macro variables are sometimes referred to as symbolic variables.

metadata

descriptive data about data that is stored and managed in a database, in order to facilitate access to captured and archived data for further use.

metadata server

a server that provides metadata management services to one or more client applications. A SAS Metadata Server is an example.

middle tier

in a SAS business intelligence system, the architectural layer in which Web applications and related services execute. The middle tier receives user requests, applies business logic and business rules, interacts with processing servers and data servers, and returns information to users.

object spawner

a program that instantiates object servers that are using an IOM bridge connection. The object spawner listens for incoming client requests for IOM services. When the spawner receives a request from a new client, it launches an instance of an IOM server to fulfill the request. Depending on which incoming TCP/IP port the request was made on, the spawner either invokes the administrator interface or processes a request for a UUID (Universal Unique Identifier).

plug-in

a file that modifies, enhances, or extends the capabilities of an application program. The application program must be designed to accept plug-ins, and the plug-ins must meet design criteria specified by the developers of the application program.

primary key

a column or combination of columns that uniquely identifies a row in a table.

response

the reaction that an individual has to a campaign, such as requesting a quote, making an inquiry, opening an e-mail message, or buying the product.

SAS data set

a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views.

SAS Management Console

a Java application that provides a single user interface for performing SAS administrative tasks.

SAS Metadata Repository

a container for metadata that is managed by the SAS Metadata Server.

schema

a map or model of the overall data structure of a database. A schema consists of schema records that are organized in a hierarchical tree structure. Schema records contain schema items.

spawner

See object spawner

Index