TIBCO Business Studio™ ActiveMatrix® Decisions Add-in

User's Guide

Software Release 1.3
November 2015

Two-Second Advantage®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO ActiveMatrix Administrator, TIBCO ActiveMatrix, TIBCO Business Studio, and TIBCO BusinessEvents are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2004-2015 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Preface

This guide describes the TIBCO Business Studio™ ActiveMatrix® Decisions Add-in.

Topics

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Overview

This chapter provides an overview of the TIBCO Business Studio™ ActiveMatrix® Decisions Add-in and how to use it to model decisions services.

Topics

Introduction

The TIBCO Business Studio ActiveMatrix Decisions Add-in allows you to develop a **decisions service** using TIBCO Business Studio.

A decisions service can be used to make a decision based on data passed into the service. It then returns a result to the calling application/process. The following is a very simple use case:

XYZ Healthcare Insurance is implementing a new policy insurability system. Applicants fill out an application on-line where they are given immediate feedback about their insurability.

The applicant enters the following information:

- Age
- Gender
- Whether or not they smoke
- Previous conditions
- Previous claims

The company wants to implement the insurability check as a service that can be re-used at multiple locations, as well as lifecycle the rules independent of the business process.

The service evaluates the entered data, and returns one of four risk factors (0-3), where 0 means the applicant is not qualified for a policy. The applicant is immediately told if they are not qualified. Applicants with a risk factor of 3 are immediately told that they are accepted for a policy. Whereas, applicants with risk factors 1 and 2 are told that their application requires further review, in which case the application is routed to a specialist to determine insurability.

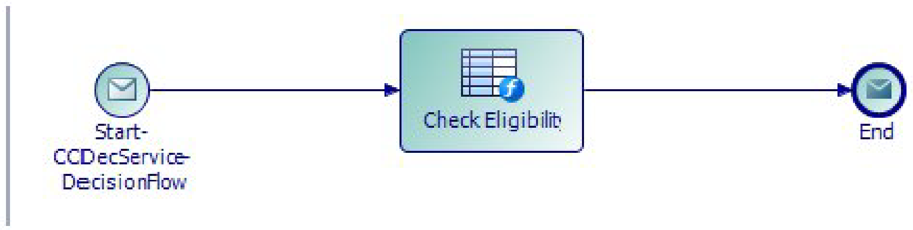
A decisions service can be created using TIBCO Business Studio - BPM Edition. To be able create a decisions service with TIBCO Business Studio, however, you must also install the TIBCO Business Studio ActiveMatrix Decisions Add-in.

Exposing a Decisions Service

A decisions service can be designed and configured to either be public or private, as follows:

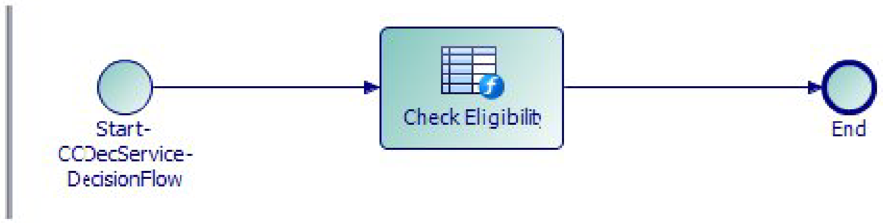
- **Public** - This exposes the decisions service as a *web service*. The web service can be called from either a client application (using the WSDL that is automatically generated), or from a *Decisions Service Task* inside of a BPM process.

  A decisions service is made public by configuring the start event to have a *trigger type* of "Message", and the end event to have a *result type* of "Message" (which is signified by the start and end events containing an envelope icon):



- **Private** - This means that the decisions service can *only* be called from a *Decisions Service Task* inside of a BPM process — a WSDL is *not* created for the decisions service, and it is not publicly exposed.

  A decisions service is made private by configuring the start event to have a *trigger type* of "None", and the end event to have a *result type* of "None" (which is signified by the start and end events not containing an icon):



How to configure the start and end events is discussed in more detail later in this guide.

Elements of a Decisions Service

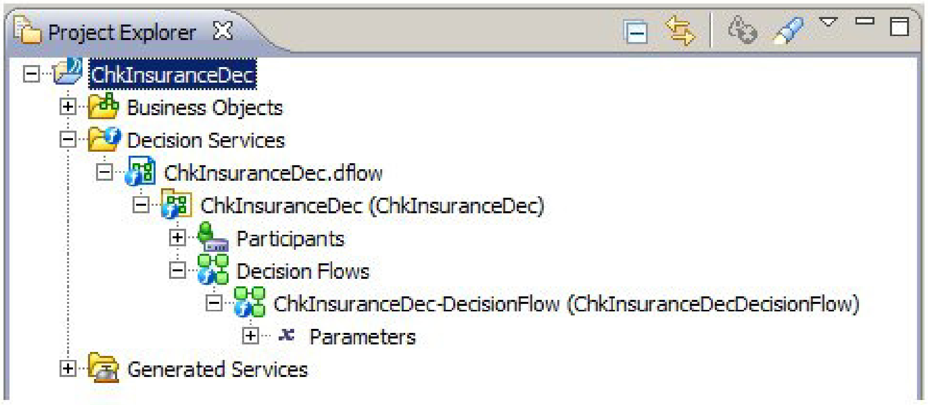
The following describes the elements of a decisions service:

- **Decisions service project** - The top-level container in TIBCO Business Studio that contains other decisions service-related elements.
- **Decisions flow package** - A sub-container within a decisions project that contains one or more *decisions flows* (see below).
- **Decisions flow** - The model of the decisions service, which includes a series of one or more *decisions table tasks.*
- **Decisions table task** - A single step in a decisions flow, which is bound to a *decisions table* (see below).
- **Decisions table** - Contains one or more *rules* (see below).
- **Rule** - A row in a decisions table. Each rule contains one or more *conditions* (which evaluate input data), and one or more *actions* (which act upon the input data to produce a result for the decisions service).
- **Business Object Model (BOM)** - Defines the data types that can be passed into the decisions service, and returned by the decisions service.

These are described in more detail in the following subsections.

Decisions Service Project

When a new project is created in TIBCO Business Studio, you can specify that it be a **Decisions Services Project**. It contains the elements of a decisions service:



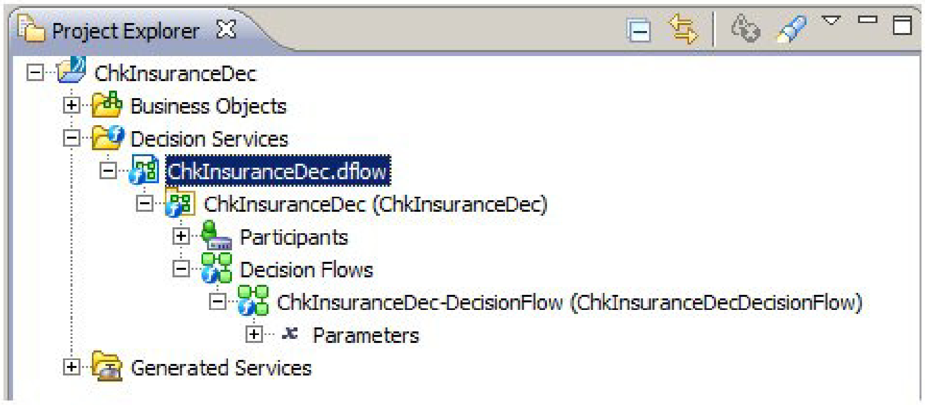
In this example, "ChkInsuranceDec" is the name of the decisions service project.

Note that in this example, the BOM is inside the project — it can also be external from the decisions service project.

For information about creating a decisions service project, see Creating a Decisions Service Project on page 16.

Decisions Flow Package

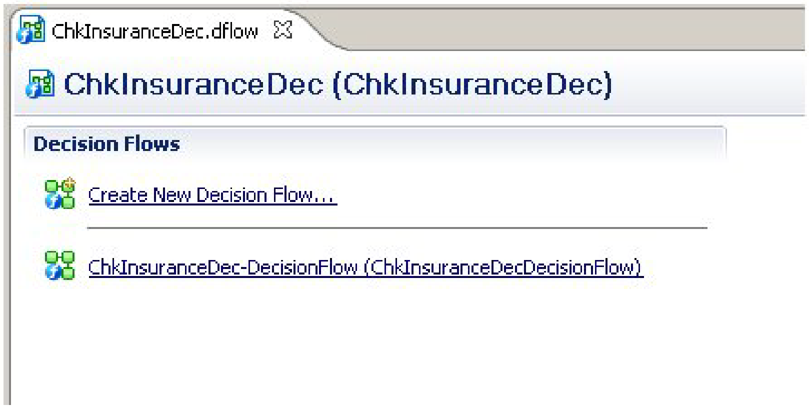
When a decisions service project is created, a *decisions flow package* is added to project:



By default, the decisions flow package is given the same name of the decisions service project, with a **.dflow** extension.

The decisions flow package can contain one or more decisions flows.

If you double-click on a decisions flow package in Project Explorer, a view similar to the following is displayed, which provides a **Create New Decision Flow** link used to add an additional decisions flow to the package, as well as links that display each of the current decisions flows in the package:


ChkInsuranceDec.dflow

ChkInsuranceDec (ChkInsuranceDec)

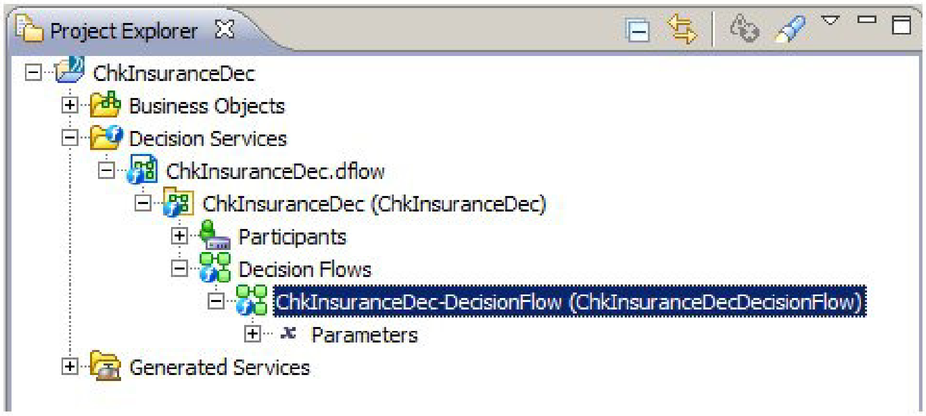
Decision Flows

Create New Decision Flow...

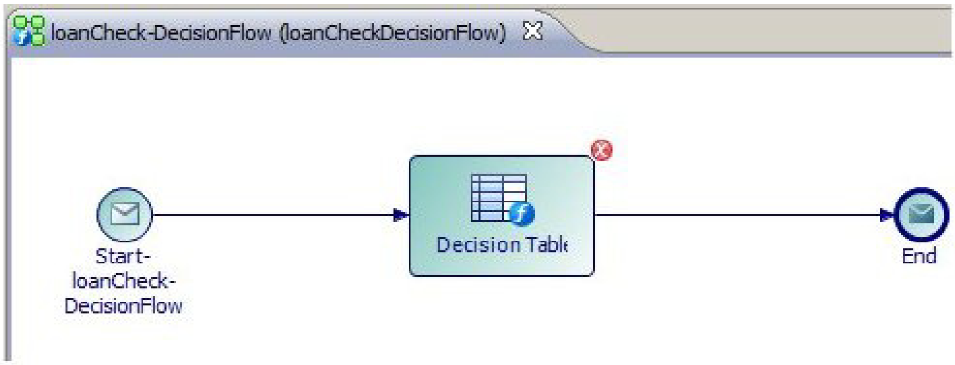
ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)

Decisions Flow

A decisions flow is a model of the decisions service. One decisions flow is created by default in each decisions flow package, inside the **Decision Flows** folder:


```
Project Explorer
ChkInsuranceDec
  Business Objects
  Decision Services
    ChkInsuranceDec.dflow
      ChkInsuranceDec (ChkInsuranceDec)
        Participants
        Decision Flows
          ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)
            Parameters
  Generated Services
```

Double-clicking on the decisions flow in Project Explorer displays the decisions flow in the decisions flow editor. For example:


loanCheck-DecisionFlow (loanCheckDecisionFlow)
Start-loanCheck-DecisionFlow
Decision Table
End

A decisions flow can include one or more *decisions table tasks*, each of which is bound to a *decisions table* containing *rules*.

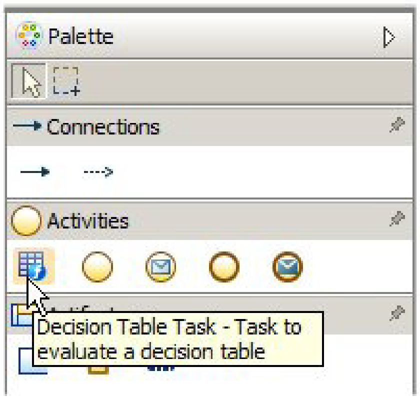
If a decisions flow contains multiple decisions table tasks, they are always executed in series — there can be no branching, conditions, looping, etc., in a decisions flow.

You can have multiple decisions flows within a single decisions flow package. Each decisions flow can be individually executed after deployment, either from a client application and/or from within a BPM process (depending on how the decisions flow is configured). If there are multiple decisions flows, each is a separate *operation* in the generated WSDL, allowing each decisions flow to be separately executed.

For information about creating additional decisions flows, see Adding a Decisions Flow to a Decisions Package on page 20.

Decisions Table Task

The decisions table task is the only task type that is available to add to a decisions flow. By default, a decisions flow contains a single decisions table task. You can add additional decisions table tasks by using the **Decision Table Task** icon in the Palette when the decisions flow editor is displayed:

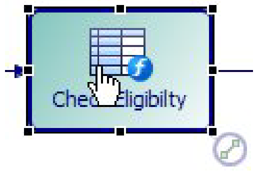


Decision Table Task - Task to evaluate a decision table

If multiple decisions table tasks are added to a decisions flow, they are always executed in sequence, from left to right.

You can edit a decisions table task by either double-clicking on the task icon in the decisions flow, or by selecting the decisions table task, then clicking the **Edit Decision Table** button on the **General** tab of the **Properties** view. The decisions table task editor is displayed — see Decisions Table on page 9.

Note, however, if you double-click the decisions table task icon to open the editor, you must place the mouse pointer over the icon in such a way that the pointer changes to a hand before double clicking:

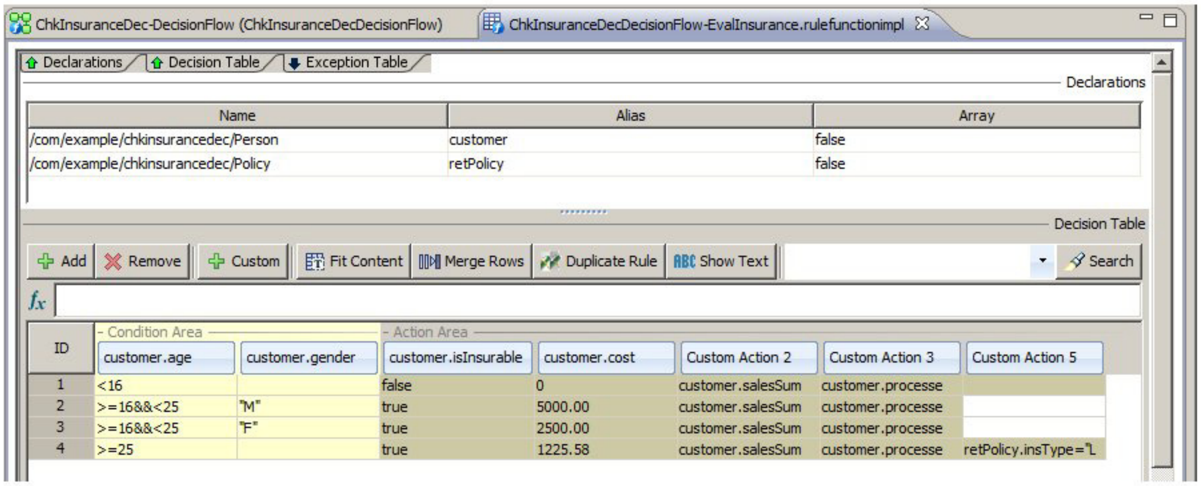


For information about adding decisions table tasks to a decisions flow, see Adding Decisions Table Tasks to a Decisions Flow on page 33.

Decisions Table

The decisions table defines rules that evaluate the data passed into the decisions service, and determines the data returned from the decisions service.

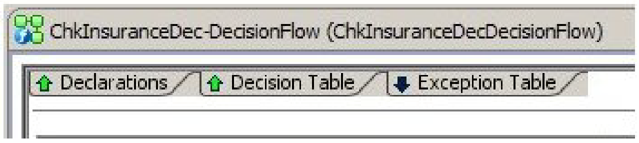
You can view/edit a decisions table by either double-clicking on a decisions table task icon in a decisions flow, or by selecting the decisions table task, then clicking the **Edit Decision Table** button on the **General** tab of the **Properties** view. The decisions table editor is displayed:



Name	Alias	Array
/com/example/chkinsurancedec/Person	customer	false
/com/example/chkinsurancedec/Policy	retPolicy	false

ID	customer.age	customer.gender	customer.isInsurable	customer.cost	Custom Action 2	Custom Action 3	Custom Action 5
1	<16		false	0	customer.salesSum	customer.processe	
2	>=16&&<25	"M"	true	5000.00	customer.salesSum	customer.processe	
3	>=16&&<25	"F"	true	2500.00	customer.salesSum	customer.processe	
4	>=25		true	1225.58	customer.salesSum	customer.processe	retPolicy.insType="L

The decisions table editor consists of three sections that can be displayed in the editor by clicking on the **Declaration**, **Decision Table**, and **Exception Table** tabs at the top of the editor. Any or all of the sections can be displayed at one time — a green up arrow indicates the section is currently displayed; a black down arrow indicates the section is not currently displayed:



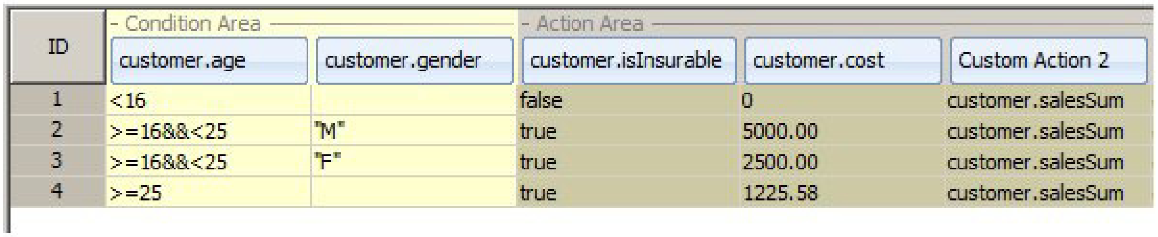
For information about these sections in the decisions table editor, see Decisions Table Task Editor on page 44.

Rules

Rules contain *conditions*, which allow you to evaluate data passed into the decisions service, and *actions*, which determine the data that is returned by the decisions service.

Each row of the decisions table constitutes a rule. The decisions table can have one or more rules. And each rule can have multiple conditions and multiple actions, which can be added to the **Condition Area** and **Action Area**.

Using the following example, these rules look at the values in **customer.age** and **customer.gender**, and set the values in **customer.isInsurable** and **customer.cost**, accordingly:



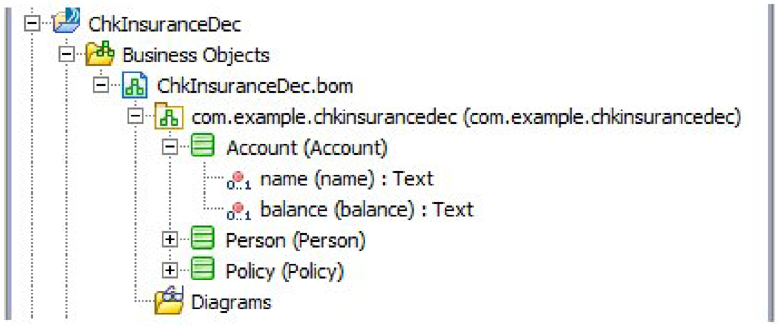
ID	Condition Area		Action Area		
	customer.age	customer.gender	customer.isInsurable	customer.cost	Custom Action 2
1	<16		false	0	customer.salesSum
2	>=16&&<25	"M"	true	5000.00	customer.salesSum
3	>=16&&<25	"F"	true	2500.00	customer.salesSum
4	>=25		true	1225.58	customer.salesSum

Also notice that *custom conditions* and *custom actions* can be included in a rule. Custom conditions and actions allow you to use operators, functions, and script to define more complex statements.

For more information about defining rules, see Defining Rules on page 49.

Business Object Model (BOM)

All data types used in a decisions service must be defined in a Business Object Model (BOM). The BOM can either be included in the same project as the decisions service, or outside of it (but in the same workspace).

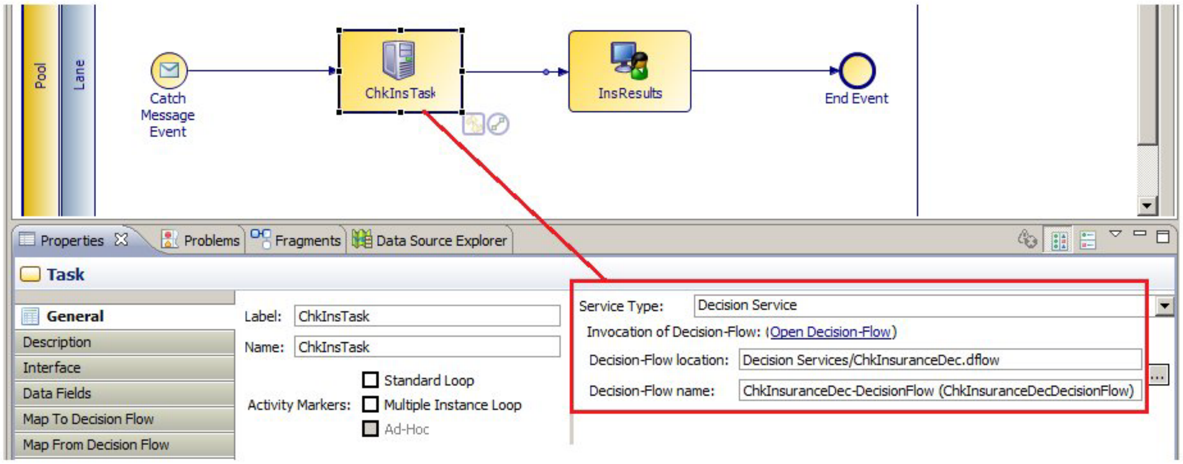


```
ChkInsuranceDec
  Business Objects
    ChkInsuranceDec.bom
      com.example.chkinsurancedec (com.example.chkinsurancedec)
        Account (Account)
          name (name) : Text
          balance (balance) : Text
        Person (Person)
        Policy (Policy)
      Diagrams
```

For information about adding a BOM to your project, see Adding a Business Object Model to a Decisions Project on page 23.

Decisions Service Task

A decisions service task is a sub-type of service task that can be added to a BPM process. This allows a decisions service to be called from within the BPM process. The properties of the decisions service task specify the name (decisions flow) and location (decisions flow package) to invoke when that task is reached in the process. The following example shows a decisions service task in a BPM process that calls decisions flow, **ChkInsuranceDecDecisionFlow**, in the decisions flow package, **ChkInsuranceDec.dflow**:



Label: ChkInsTask
Name: ChkInsTask
Activity Markers: Standard Loop, Multiple Instance Loop, Ad-Hoc
Service Type: Decision Service
Invocation of Decision-Flow: (Open Decision-Flow)
Decision-Flow location: Decision Services/ChkInsuranceDec.dflow
Decision-Flow name: ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)

The decisions flow can be opened from the BPM process by clicking on the **Open Decision-Flow** link on the **General** tab for the decisions service task**.**

For information about adding and configuring a decisions service task in a BPM process, see Calling a Decisions Service from a BPM Process, page 77.

Decisions Service Deployment

After a decisions service project is complete, it needs to be deployed to a runtime environment. This can be done in the following ways:

- Directly from TIBCO Business Studio
- Export or generate the decisions project as a Distributed Application Archive (DAA), then deploy using TIBCO ActiveMatrix® Administrator
- By generating a deployment script, then invoking the script to deploy to the desired node.
- Using the Generate Administrator CLI Script wizard.

For more information, see Decisions Service Deployment on page 97.

Samples

When the TIBCO Business Studio ActiveMatrix Decisions Add-in is installed, a number of sample TIBCO Business Studio projects are included. These samples, which provide examples of decisions services, can be imported into your TIBCO Business Studio workspace.

The samples are installed in the following location:

*STUDIO_HOME***studio****3**.n**sda****samples****sda_samples**

The sample projects include:

- **Deployment** - The Deployment folder under sda_samples includes examples of generating DAAs from a command line for decisions projects, as well as creating scripts to deploy decisions projects using the command line interface (CLI):
  - — CL-Build-DAA - This folder contains an example of generating DAAs from a command line for all of the decisions sample projects.
  - — CL-Build-DAA-output - This folder contains the output from the execution of the DAA-generation scripts found in the CL-Build-DAA folder.
  - — DeploySamples - This folder contains a deployment project for generating CLI scripts that can be used to deploy all of the decisions sample and tutorial projects. You can import this deployment project into your TIBCO Business Studio workspace and view the configuration needed to generate CLI scripts.

  For information about generating DAAs from a command line for decisions projects, see Generating a DAA From the Command Line on page 102.

  For information about creating scripts to deploy decisions projects using the command line interface, see Deployment Scripts for Decisions Projects on page 105.
- **Sample-Array** - Provides an example of using arrays in decisions tables.
- **Sample-DateTime** - Shows the use of the Date, Time, DateTime, and DateTimeTZ types.
- **Sample-NewInstance** - Demonstrates the use of the **newInstance** standard function to create new instances of objects.
- **Sample-Priority** - Illustrates the use of priorities to control the execution of rules.
- **Sample-GlobalData** - Illustrates the use of global data in a decisions project.

Chapter 2 Creating a Decisions Service Project

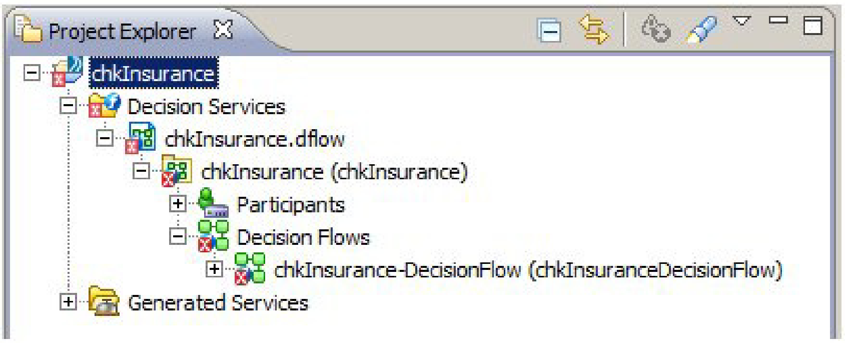
This chapter describes how to create a decisions service project.

Topics

Creating a Decisions Service Project

TIBCO Business Studio allows you to create a decisions service inside of a project that includes other assets, such as BPM processes and SOA composites, as well as BOMs and organization models.

This section describes how to create a standalone decisions service project using the wizard provided in TIBCO Business Studio.



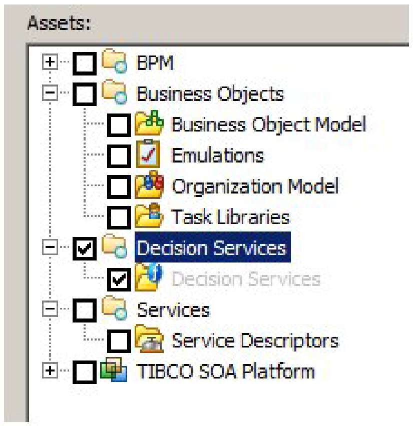
To create a new decisions service project:

1. Start TIBCO Business Studio and specify a new workspace in which your decisions service project and its assets will be saved.
2. Do either one of the following:
   a. Select **File** > **New** > **Other > Decision Services Project**.
   b. Right-click in Project Explorer, then select **New** > **Decision Services Project**.
3. On the **New Decision Services Project** dialog, enter a name the **Project name** field.
4. In the **Destination Environments** section, check the appropriate boxes as follows, then click **Next**:
   — **Decisions** - This box must be checked.
   — **BPM** - This must be checked if the decisions service is going to be called from a *decisions service task* in a BPM process. (This selection is available only if using TIBCO Business Studio - BPM Edition.)

   The **Asset Type Selection** dialog displays, which is used to specify all of the assets that you want included in your project. For instance, you can include the BOM inside of the project (or a BOM can be created separately outside of your decisions project, which allows the BOM to be referenced by multiple projects).

The **Asset Type Selection** dialog also allows you to choose other TIBCO products, such as BPM and TIBCO SOA Platform. This procedure assumes that only a decisions service project is being created.

5. Select the appropriate boxes on the **Asset Type Selection** dialog, then click **Next**.



The **Decision Services** dialog displays, which is used to modify the name of the decisions flow package folder in which your decisions flow(s) are stored, as well as the name of the default decisions flow that will be created in that folder.

6. Modify, if desired, the name of the decisions flow package folder and/or name of the default decisions flow (it must have a **.dflow** extension) that will be created in the package.

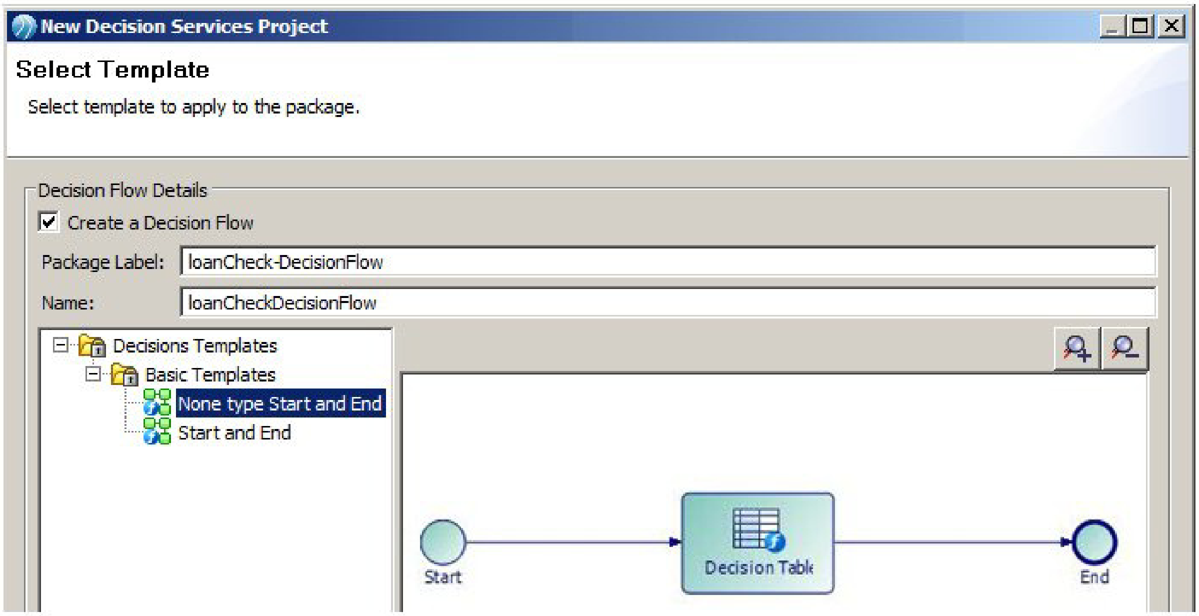
   You can also uncheck the **Create Decision Flow Package** check box if you do not want a default decisions flow created in the package. (For information about adding a decisions flow at a later time, see Adding a Decisions Flow to a Decisions Package on page 20.)

   After making any desired changes on the **Decision Services** dialog, click **Next**.

   The **Package Information** dialog displays, which shows details about the decisions flow package that will be created.

7. On the **Package Information** dialog, either accept the default properties of the package, or modify them as necessary, then click **Next**.

The **Select Template** dialog displays, which allows you to select a template that determines the trigger type for the start and end events on the created decisions flow.


New Decision Services Project

Select Template

Select template to apply to the package.

Decision Flow Details

☑ Create a Decision Flow

Package Label: loanCheck-DecisionFlow

Name: loanCheckDecisionFlow

8. Select the desired template, as follows:

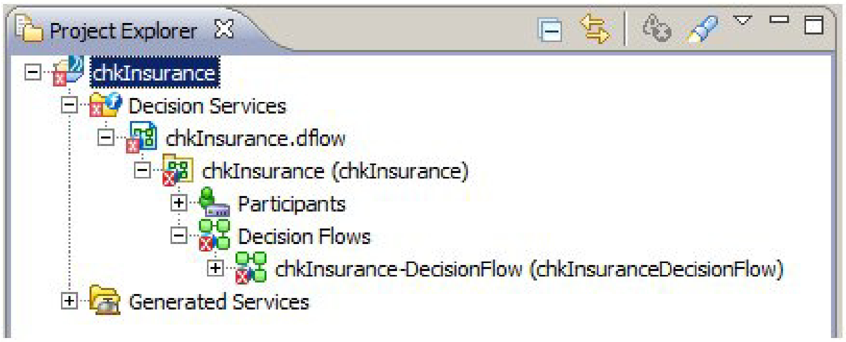
   — **None type Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "None". This makes the decisions service private (that is, it is not publicly exposed as a web service) — it can *only* be called from a decisions service task inside of a BPM process. When this template is chosen, a WSDL is *not* created in the **Generated Services** folder in Project Explorer.

   — **Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "Message". When this template is chosen, a WSDL is created in the **Generated Services** folder in Project Explorer, allowing the decisions service to be publicly exposed and consumed as a web service.

   After selecting the desired template, click **Next**.

   The remainder of the dialogs displayed in the **New Decision Services Project** wizard depend on selections you made on the **Asset Type Selection** dialog in step 5. The defaults can be accepted on all of these dialogs, or you can make modifications as desired (for details about those dialogs, see the *TIBCO Business Studio Process Modeling User's Guide*).

After you have completed the dialogs in the wizard, the newly-created project, package, and decisions flow are displayed in the Project Explorer.



The problem marker on the newly created project is because the default decisions table in the decisions flow is empty.

While trying to create a decision project on a Unix system, you may see the error "Failed to complete creation of project<*project_name*>".

Workaround: In the machine's *etc/hosts* file, add the following entry, substituting the machine specific details:

<*your-machine-ip*> <*your-machine-name*> localhost

For example,

10.97.109.168 bpmrhel764bit localhost

For information about defining the decisions table, see Working with Decisions Flows on page 31.

Adding a Decisions Flow to a Decisions Package

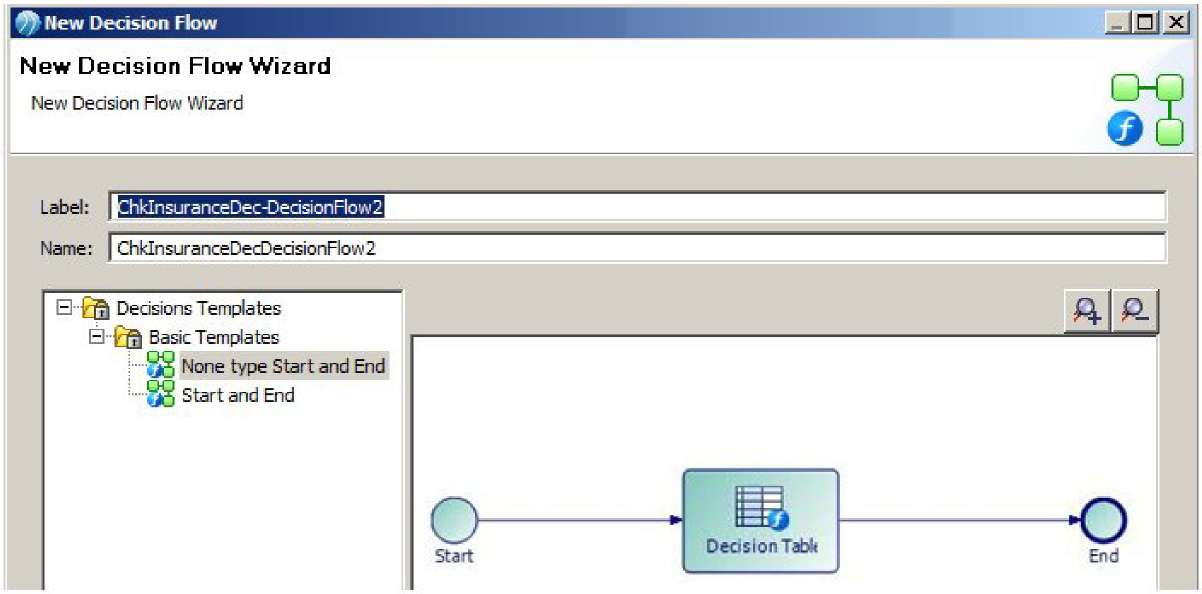
After a decisions service project is created with the wizard (see Creating a Decisions Service Project on page 16), you can add additional decisions flows to the package.

Each decisions flow in a package can be individually executed after deployment, either from a client application and/or from within a BPM process (depending on how the decisions flow is configured). If there are multiple decisions flows, each is a separate *operation* in the generated WSDL, allowing each decisions flow to be separately executed.

To add a decisions flow to a package, do the following:

1. In the TIBCO Business Studio Project Explorer, right-click on the decisions package (which has the **.dflow** extension) in an existing decisions project, and select **New** > **Decision Flow**.

   The **New Decision Flow Wizard** displays.



New Decision Flow

New Decision Flow Wizard

New Decision Flow Wizard

Label: ChkInsuranceDec-DecisionFlow2

Name: ChkInsuranceDecDecisionFlow2

   By default, the label and name given to the new decisions flow are the same as the existing decisions flow, with a "2" appended.

2. Modify the label and name of the new decisions flow, if desired, then select the template you would like to use, as follows:

   — **None type Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "None". This makes the decisions service private (that is, it is not publicly exposed as a web service) — it can *only* be called from a decisions service task inside of a BPM process. When this template is chosen, a WSDL is *not* created in the **Generated Services** folder in Project Explorer.

   — **Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "Message". When this template is chosen, a WSDL is created in the **Generated Services** folder in Project Explorer, allowing the decisions service to be consumed as a web service.

   After selecting the desired template, click **Next**.

3. On the **Description** dialog, add optional text that describes the decisions flow, an optional URL that links to documentation about the decisions flow, and click **Next**.

   The **Documentation Url** field is intended for design-time collaboration; it is not displayed in the runtime environment.

4. On the **Destinations** dialog, specify the destination of the new decisions flow, as follows:

   — **Decisions** - This box must be checked

   — **BPM** - This must be checked if the decisions service is going to be called from a *decisions service task* in a BPM process.

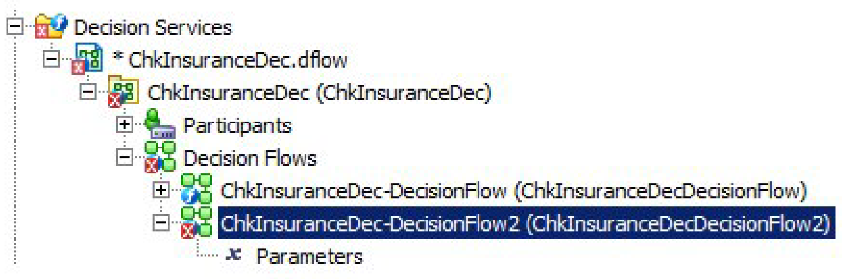
   After checking the appropriate box(es), click **Next**.

   The specific destination components that make up a destination environment can be viewed by selecting **Window** > **Preferences**, and selecting **Destination Environments**.

   The **Extended** dialog displays, which allows you to add optional supplemental information to the decisions flow. For more information see the *TIBCO Business Studio Process Modeling* guide.

5. Click **Finish**.

The new decisions flow is added to the decisions flow package:


Decision Services
* ChkInsuranceDec.dflow
ChkInsuranceDec (ChkInsuranceDec)
Participants
Decision Flows
ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)
ChkInsuranceDec-DecisionFlow2 (ChkInsuranceDecDecisionFlow2)
Parameters

The problem marker on the new decisions flow is because the new flow has an empty decisions table.

The new decisions flow is also opened in the decisions flow editor.

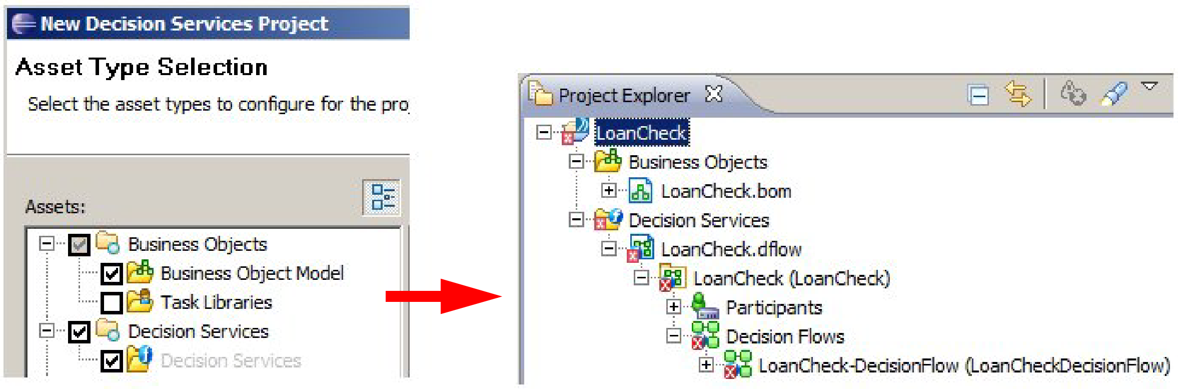
For information about working with decisions flows, see Working with Decisions Flows on page 31.

Adding a Business Object Model to a Decisions Project

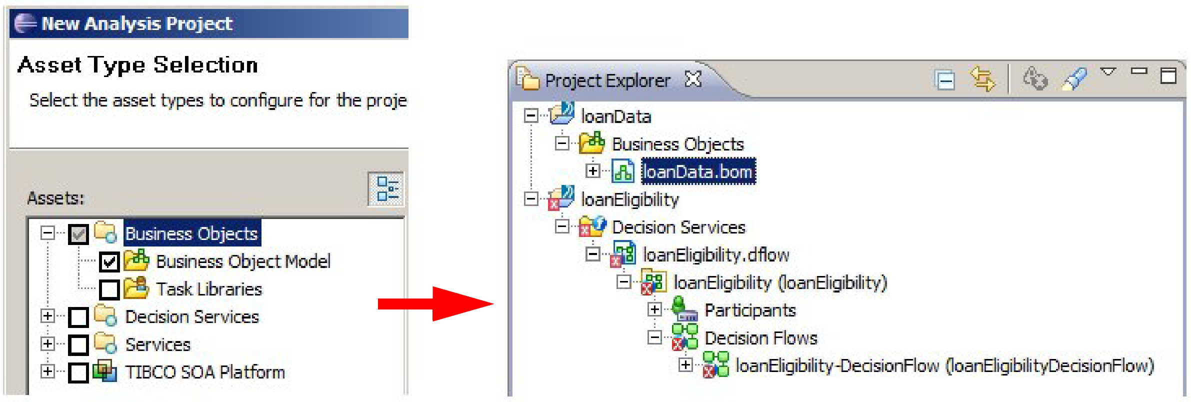
All parameter data that is input to a decisions flow, or output from a decisions flow, must be defined in a Business Object Model (BOM).

The BOM can either be included inside of your decisions service project or outside of it as its own project.

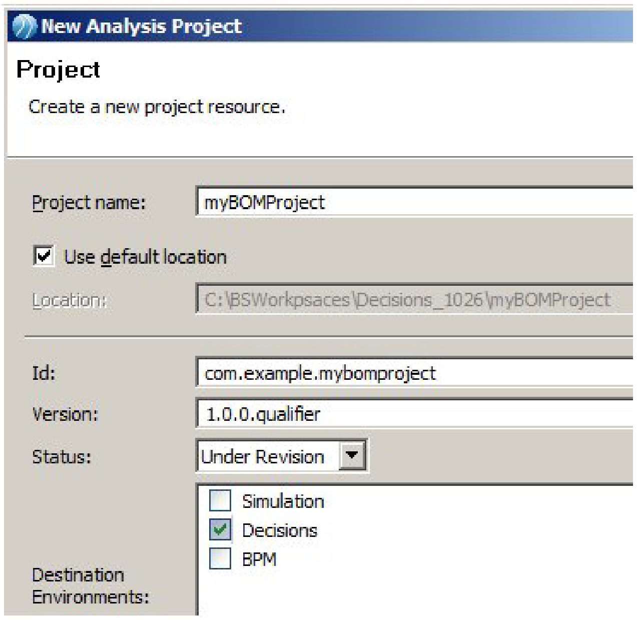
If you select both **Business Objects** and **Decision Services** on the **Asset Type Selection** dialog when creating your decisions service project, the BOM is created inside of the decisions project:



If you create the BOM as a separate project from your decisions project, it is outside of the decisions project:



If you create the BOM as a separate project, outside of your decisions services project, you must set its destination environment as "Decisions" when the project is created:


New Analysis Project

Project

Create a new project resource.

Project name: myBOMProject

☑ Use default location

Location: C:\BSWorkpsaces\Decisions_1026\myBOMProject

Id: com.example.mybomproject

Version: 1.0.0.qualifier

Status: Under Revision

Destination Environments:
☐ Simulation
☑ Decisions
☐ BPM

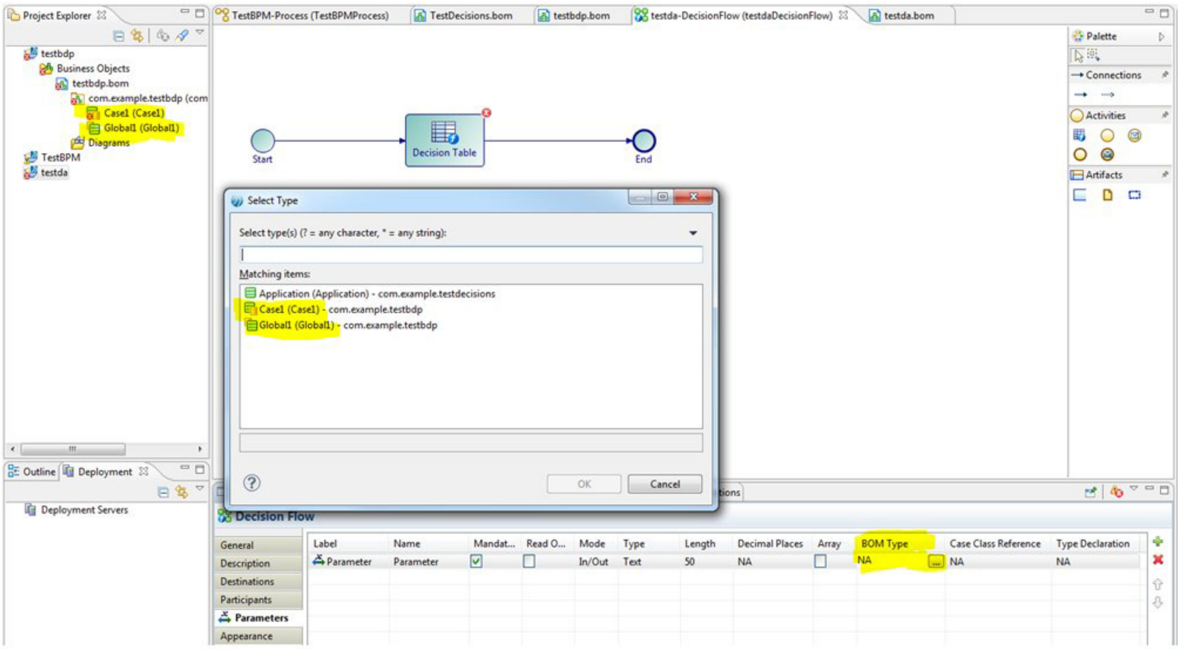
The data types that are defined in the BOM must be mapped to *formal parameters* that are defined in the decisions flow. For information about mapping the formal parameters to the BOM types, see Adding Parameters to a Decisions Flow, page 38.

For details about creating Business Object Models, and adding data types to the BOM, see the *TIBCO Business Studio Business Object Modeler* guide.

Global Data (Business Data Projects)

TIBCO ActiveMatrix BPM supports *case* and *global* data types. These data types can only be defined in B*usiness Data Projects*, and therefore must be external to decisions projects.

Global instance data is held external to the BPM process instances at runtime. ActiveMatrix Decisions does not provide direct access to global instance data, therefore it must be provided through decisions flow parameters. However, ActiveMatrix Decisions does support the use of global data types as parameter definitions.



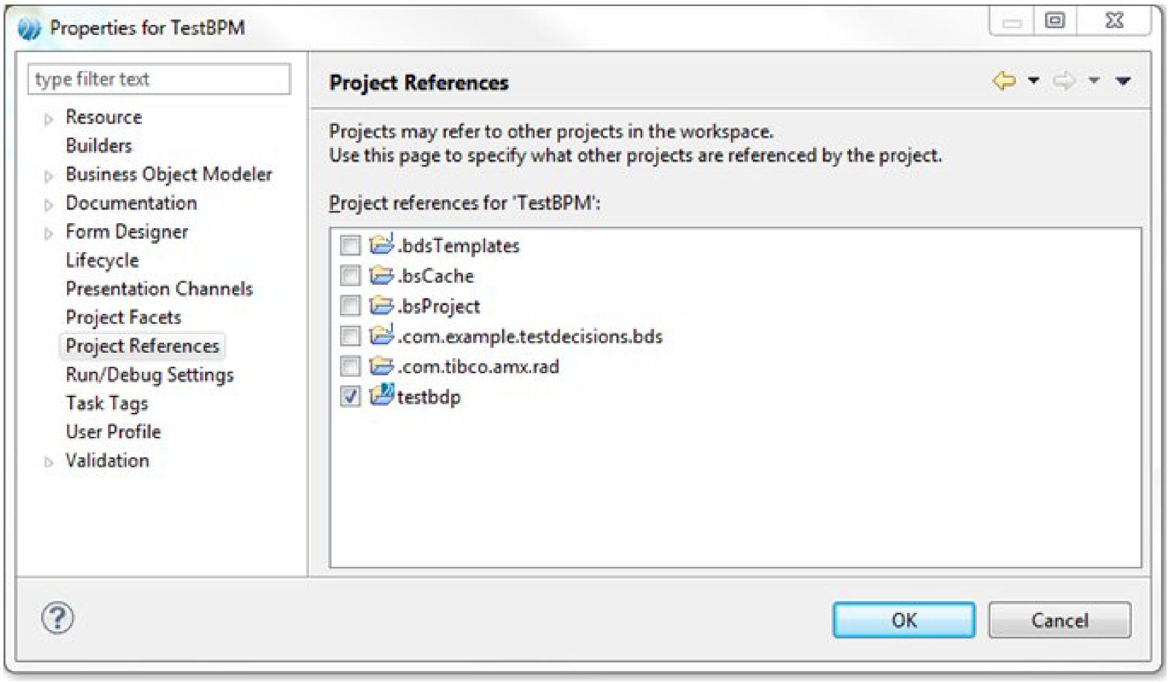
Select type(s) (? = any character, * = any string):

Matching items:

- Application (Application) - com.example.testdecisions
- Case1 (Case1) - com.example.testbdp
- Global1 (Global1) - com.example.testbdp

Label	Name	Mandat...	Read O...	Mode	Type	Length	Decimal Places	Array	BOM Type	Case Class Reference	Type Declaration
Parameter	Parameter	✓	☐	In/Out	Text	50	NA	☐	NA	NA	NA

If global data types are used as parameter definitions, the relevant Business Data Project must be included as a referenced project:



Project References

Projects may refer to other projects in the workspace.
Use this page to specify what other projects are referenced by the project.

Project references for 'TestBPM':

- ☐ .bdsTemplates
- ☐ .bsCache
- ☐ .bsProject
- ☐ .com.example.testdecisions.bds
- ☐ .com.tibco.amx.rad
- ☑ testbdp

Business Object Model Restrictions

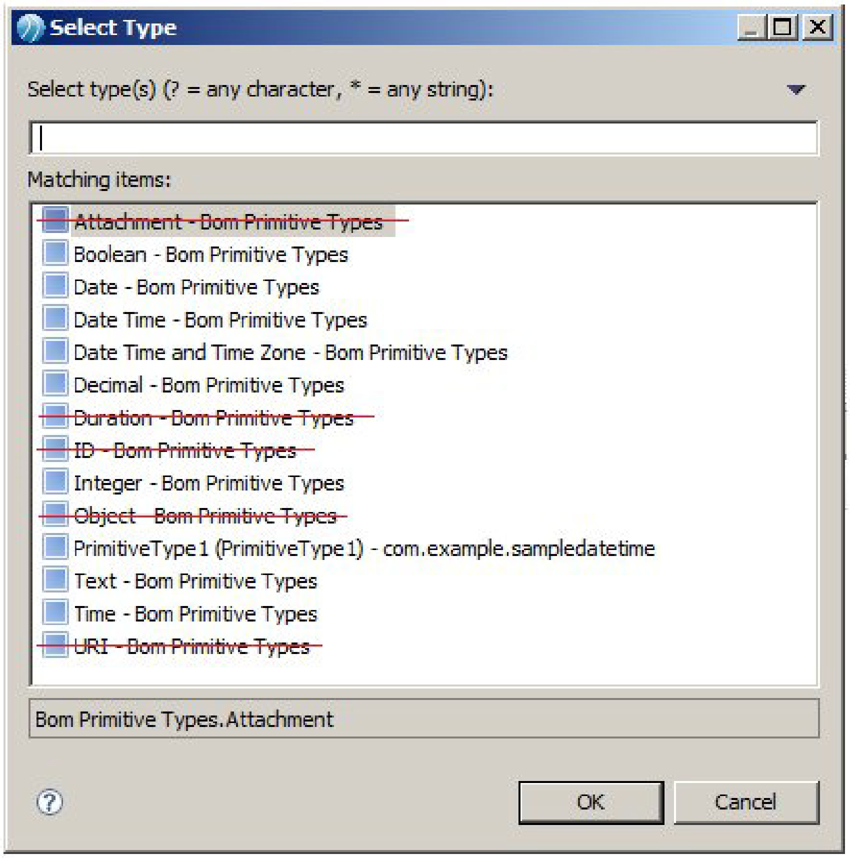
The following restrictions apply to business object models for use with decisions services:

Enumerations

Values in enumerations are *always* evaluated as strings.

Data Types

There are some BOM primitive data types that are not supported for use in decisions flow tables. You cannot use the **Attachment**, **Duration**, **ID**, **Object**, and **URI** BOM primitive types:



~~Attachment - Bom Primitive Types~~
Boolean - Bom Primitive Types
Date - Bom Primitive Types
Date Time - Bom Primitive Types
Date Time and Time Zone - Bom Primitive Types
Decimal - Bom Primitive Types
~~Duration - Bom Primitive Types~~
~~ID - Bom Primitive Types~~
Integer - Bom Primitive Types
~~Object - Bom Primitive Types~~
PrimitiveType1 (PrimitiveType1) - com.example.sampledatetime
Text - Bom Primitive Types
Time - Bom Primitive Types
~~URI - Bom Primitive Types~~

Bom Primitive Types.Attachment

There are some data types in your business object model that are stored as different data types in a decisions service. The following table describes how a decisions service stores data types at runtime.

Business Object Model Data Type	Decisions Services Data Type
Boolean	Boolean
Date	DateTime
DateTime	DateTime

Business Object Model Data Type	Decisions Services Data Type
DateTimeTz	DateTime
Decimal (Floating Point) [Represented internally as a Double]	Double
Decimal (Fixed Point) [Represented internally as a BigDecimal]	Double
Integer (Signed Integer) [Represented internally as an Integer]	Integer
Integer (Fixed Length) [Represented internally as a BigInteger]	Long
Text	String
Time	DateTime

A warning is generated in the **Problems** view that informs you if a data type is being stored as a different data type at runtime.

Dates and Times

All types that are defined in the BOM as Date or Time are evaluated as DateTime in decisions table rules. For instance, a Date BOM type is viewed in a decisions table as a DateTime, with the time portion set to 00:00:00. And a Time BOM type is viewed in a decisions table as a DateTime, with the date portion set to the epoch date (1970-01-01).

You can use the available catalog functions (such as getDate, getMonth, getHour, etc.) to manipulate dates and times, as needed. For information about the catalog functions, see Using the Standard Functions on page 65.

For examples of using dates and times, see the "Sample-DateTime" project provided in the samples that are included when you install the TIBCO Business Studio ActiveMatrix Decisions Add-in. These are included in the **ADEC_Samples.zip** file, which is located in the following directory:

*STUDIO_HOME***studio****3**.*n***sda****samples**\

Text

If you have a condition in a decisions table where either:

- the type is text, or
- it is an array whose type is text...

and you want to assign a value of **100**, for example:

- Field = 100;
- Field.arr1 = 100;

the condition will not be executed if, in a BPM process script, an integer value is passed. This is because JavaScript is unable to determine the type in this case and converts it to a double **100.0**, which leaves the decisions table condition unsatisfied.

A workaround for this is to pass the value in double quotes, for example:

```
Field = "100";
```

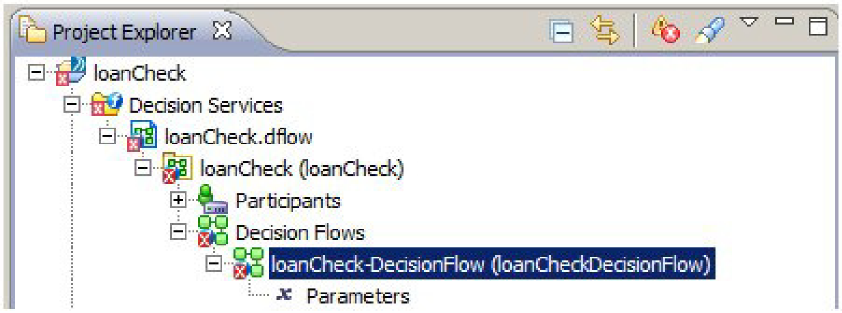
Chapter 3 Working with Decisions Flows

This chapter describes working with decisions flows.

Topics

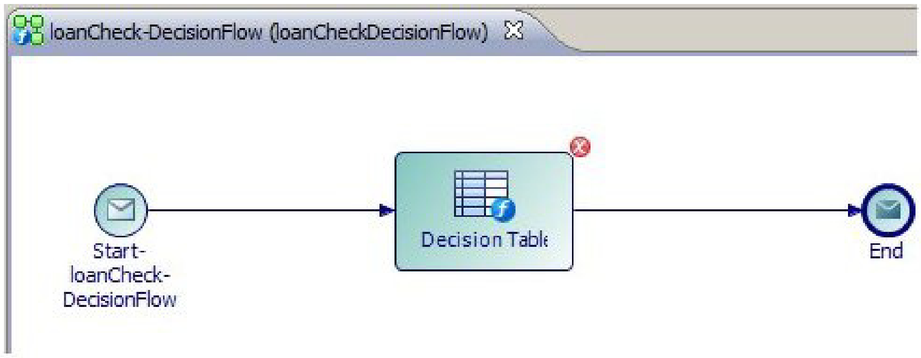
Decisions Flow Overview

When a decisions service project is created, a single decisions flow is created under the **Decision Flows** folder in Project Explorer:



Initially, the decisions flow will contain a problem marker because the default decisions table added to the decisions flow is empty.

You can double-click on the decisions flow in Project Explorer to display the decisions flow in the decisions flow editor (when the decisions project is created, the decisions flow editor opens by default). For example:


loanCheck-DecisionFlow (loanCheckDecisionFlow)
Start-loanCheck-DecisionFlow
Decision Table
End

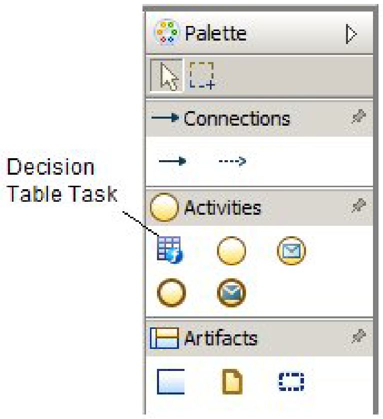
One decisions table is added to the decisions flow by default. The start and end events will contain an envelope icon if their trigger/result type had been specified as "Message" when the project was created. There is no icon shown in the start and events if their trigger/result type had been specified as "None" when the project was created. For more information about the start and end events, see Decisions Flow Start and End Events on page 34.

A decisions flow can only contain decisions table tasks that are executed in series; there can be no branching, conditions, looping, etc., in a decisions flow.

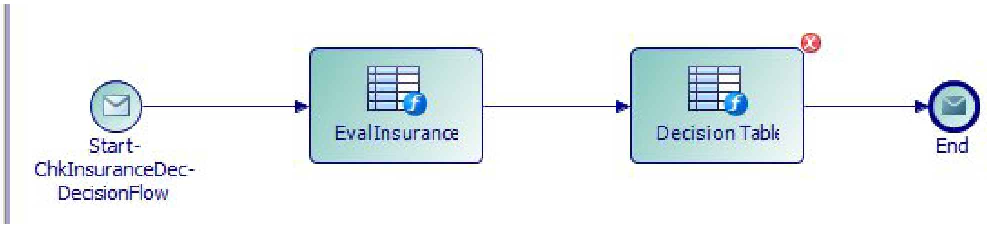
Adding Decisions Table Tasks to a Decisions Flow

A decisions flow can include one or more decisions table tasks, each of which is bound to a decisions table containing rules. When a decisions service is executed, its associated decisions flow is executed.

The palette available in the decisions flow editor contains only one type of task (besides the start and end event activities), the decisions table task:



The **Decision Table Task** activity on the palette allows you to add additional decisions tables to a decisions flow:



A newly added decisions table will have a problem marker because the decisions table is empty.

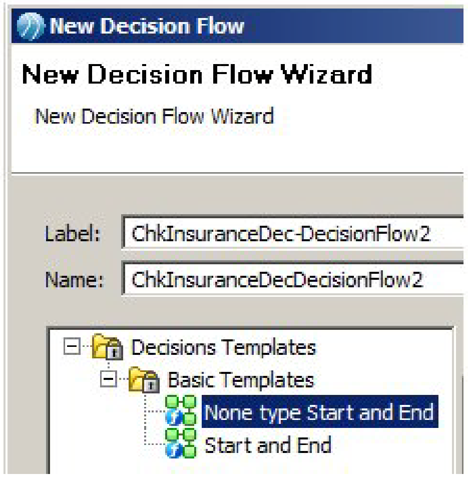
If a decisions flow contains multiple decisions table tasks, they are always executed in series — there can be no branching, conditions, looping, etc., in a decisions flow.

Parameter values that are set in a decisions table early in a decisions flow, can be evaluated by rules in decisions tables later in the decisions flow.

For information about defining the decisions tables on a decisions flow, see Working with Decisions Flows on page 31.

Decisions Flow Start and End Events

When you are creating a decisions project, or adding an additional decisions flow to an existing project, you can select a template that determines whether the decisions flow start and end events have a trigger/result type of "None" or "Message":



New Decision Flow

New Decision Flow Wizard

New Decision Flow Wizard

Label: ChkInsuranceDec-DecisionFlow2

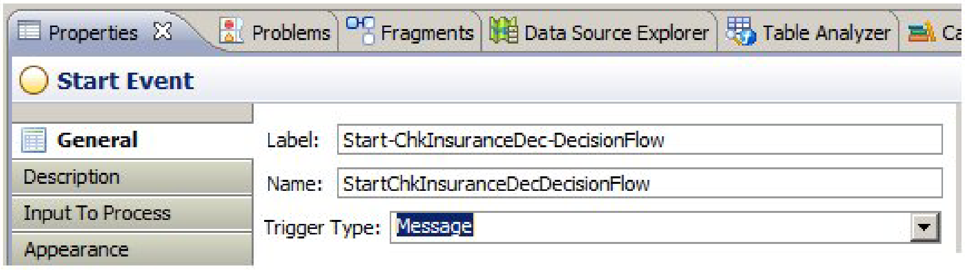
Name: ChkInsuranceDecDecisionFlow2

- Decisions Templates
  - Basic Templates
    - None type Start and End
    - Start and End

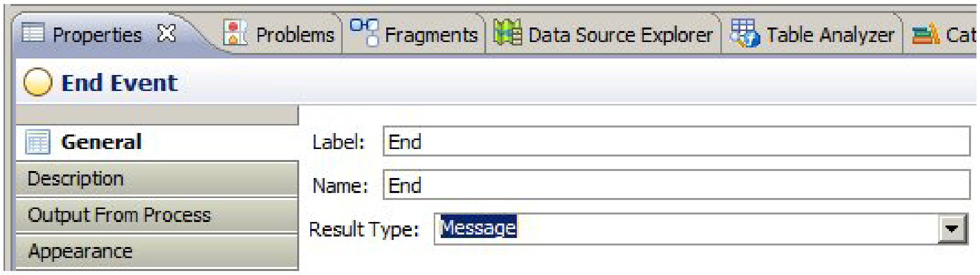
The meanings of these templates are:

- **None type Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "None". This essentially makes the decisions service private (that is, it is not publicly exposed as a web service) — it can *only* be called from a decisions service task inside of a BPM process. When this template is chosen, a WSDL is *not* created in the **Generated Services** folder in Project Explorer.
- **Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "Message". When this template is chosen, a WSDL is created in the **Generated Services** folder in Project Explorer, allowing the decisions service to be consumed as a web service.

After the decisions flow is created, you can modify the trigger/result type by selecting the start or end event in the decisions flow, then modifying the selection in the **Trigger Type** field (for the start event) or the **Result Type** field (for the end event) on the **Properties** view **General** tab:



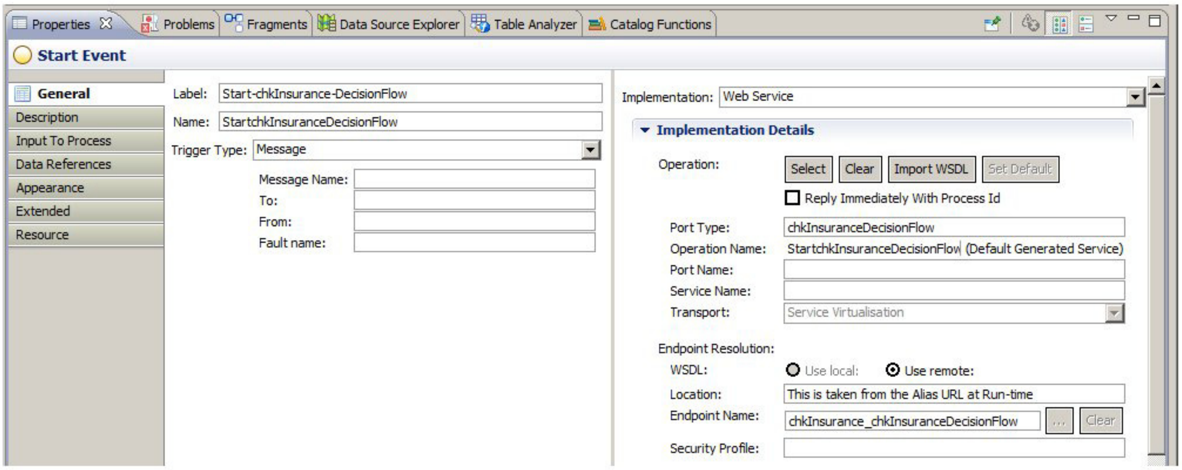
Label: Start-ChkInsuranceDec-DecisionFlow
Name: StartChkInsuranceDecDecisionFlow
Trigger Type: Message



Label: End
Name: End
Result Type: Message

"Message" Start Event Configuration

Clicking on a decisions flow start event that has been configured to have a trigger type of "Message" causes the **Properties** view **General** tab to display the configuration information for the incoming message to the decisions flow:



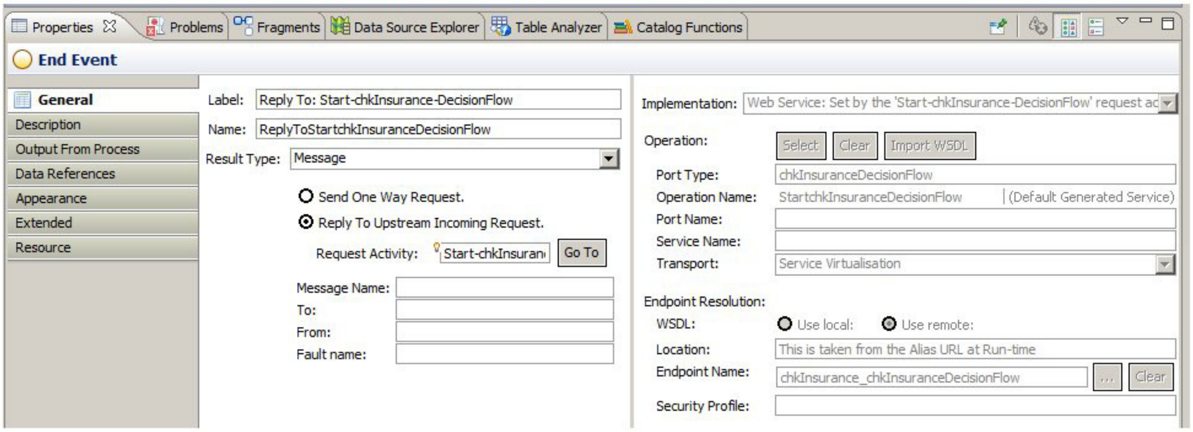
Label: Start-chkInsurance-DecisionFlow
Name: StartchkInsuranceDecisionFlow
Trigger Type: Message
Message Name:
To:
From:
Fault name:

Implementation: Web Service
Operation:
Reply Immediately With Process Id
Port Type: chkInsuranceDecisionFlow
Operation Name: StartchkInsuranceDecisionFlow (Default Generated Service)
Port Name:
Service Name:
Transport: Service Virtualisation
Endpoint Resolution:
WSDL: Use local: [✓ Use remote:]
Location: This is taken from the Alias URL at Run-time
Endpoint Name: chkInsurance_chkInsuranceDecisionFlow
Security Profile:

If you are using a WSDL to expose the decisions service, nothing on this tab should be modified. If you change the value in the **Trigger Type** field to "None", all of the web service configuration information is cleared and the decisions service becomes private.

"Message" End Event Configuration

Clicking on a decisions flow end event that has been configured to have a result type of "Message" causes the **Properties** view **General** tab to display the configuration for the outgoing message for the decisions flow:



Label: Reply To: Start-chkInsurance-DecisionFlow
Name: ReplyToStartchkInsuranceDecisionFlow
Result Type: Message
Send One Way Request.
[✓ Reply To Upstream Incoming Request.]
Request Activity: Start-chkInsuran
Message Name:
To:
From:
Fault name:

Implementation: Web Service: Set by the 'Start-chkInsurance-DecisionFlow' request ac
Operation:
Port Type: chkInsuranceDecisionFlow
Operation Name: StartchkInsuranceDecisionFlow (Default Generated Service)
Port Name:
Service Name:
Transport: Service Virtualisation
Endpoint Resolution:
WSDL: Use local: [✓ Use remote:]
Location: This is taken from the Alias URL at Run-time
Endpoint Name: chkInsurance_chkInsuranceDecisionFlow
Security Profile:

If you are using a WSDL to expose the decisions service, nothing on this tab should be modified. If you change the value in the **Trigger Type** field to "None", all of the web service configuration information is cleared and the decisions service becomes private.

Adding Parameters to a Decisions Flow

All parameter data that is input to a decisions flow, or output from a decisions flow, must be defined in a Business Object Model (BOM). For information about adding a BOM to a decisions project, see Adding a Business Object Model to a Decisions Project, page 23.

The data types that are defined in the BOM must then be mapped to *formal parameters* that are defined in the decisions flow. This section describes how to add formal parameters to a decisions flow, then map them to the BOM data types.

Note that only *top-level* parameter mapping is allowed, that is, BOM types can be mapped to formal parameters; you cannot map BOM type *attributes* to formal parameter *attributes*.

Adding Parameters to a Decisions Flow

There are two ways to add parameters to a decisions flow:

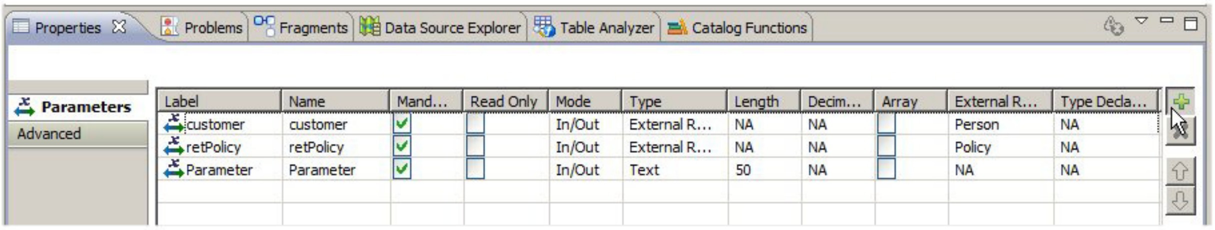
- in the **Properties** view table provided when the **Parameters** folder is selected in Project Explorer (this method is better if you need to create several parameters), or
- using the wizard to create one at a time.

Using the Properties View

To create a new parameter in the **Properties** view, do the following:

1. In the Project Explorer, select the decisions flow.
2. In the **Properties** view, select the **Parameters** tab.
3. On the **Parameters** tab, click the button containing the Plus sign .

   A new row is added to the table containing "Parameter" as the name and label:



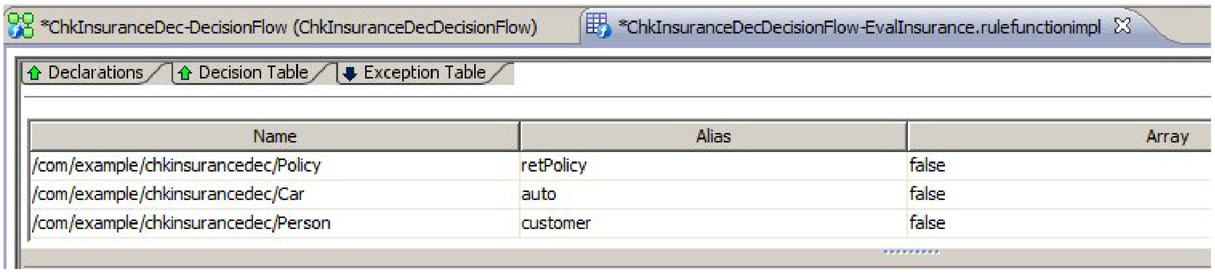
Label	Name	Mand...	Read Only	Mode	Type	Length	Decim...	Array	External R...	Type Decla...
customer	customer	✓	☐	In/Out	External R...	NA	NA	☐	Person	NA
retPolicy	retPolicy	✓	☐	In/Out	External R...	NA	NA	☐	Policy	NA
Parameter	Parameter	✓	☐	In/Out	Text	50	NA	☐	NA	NA

4. Modify each cell in the table for the new parameter, as follows:
   - **Label** - The label for the parameter. This can be different than the name.
   - **Name** - The name of the parameter. This is the value shown in the **Alias** column of the **Declarations** section in the decisions flow editor (see below). It is also the name of the parameter that you will use in the rules defined in the decisions table.
   - **Mandatory** - All parameters defined in a decisions flow must be mandatory.
   - **Read Only** - Not supported.
   - **Mode** - Specifies whether the parameter is input to the decisions service (**In**), output from the decisions service (**Out**), or input to *and* output from the decisions service (**In/Out**).
   - **Type** - This must be set to **External Reference**. Only parameters that reference a BOM type are supported in a decisions flow.
   - **Length** - Not applicable.
   - **Decimal Places** - Not applicable.
   - **Array** - Always false. For more information, see Using Arrays in Conditions and Actions on page 61.
   - **BOM Type** - Clicking in this cell allows you to choose the BOM data type that this formal parameter is referencing.
   - **Case Class Reference** - Not applicable.
   - **Type Declaration** - Not applicable.

   The newly created parameter is shown in the **Parameters** tab.

   You can also delete, as well as move parameters up and down in the table, by using the additional buttons to the right of the parameter table.

   When a parameter is added to a decisions flow, that parameter is automatically added to the **Declarations** section in the decisions flow editor:


*ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)
*ChkInsuranceDecDecisionFlow-EvalInsurance.rulefunctionimpl
Declarations
Decision Table
Exception Table

Name	Alias	Array
/com/example/chkinsurancedec/Policy	retPolicy	false
/com/example/chkinsurancedec/Car	auto	false
/com/example/chkinsurancedec/Person	customer	false

The columns in the **Declarations** sections are:

- **Name** - The external reference to the BOM data type. All parameters in a decisions flow must reference a BOM data type.
- **Alias** - The name given to the parameter when it is added to the decisions flow. This is the name that is referenced in the rules in the decisions table.
- **Array** - This is always false, as formal parameters cannot be arrays. For more information, see Using Arrays in Conditions and Actions on page 61.

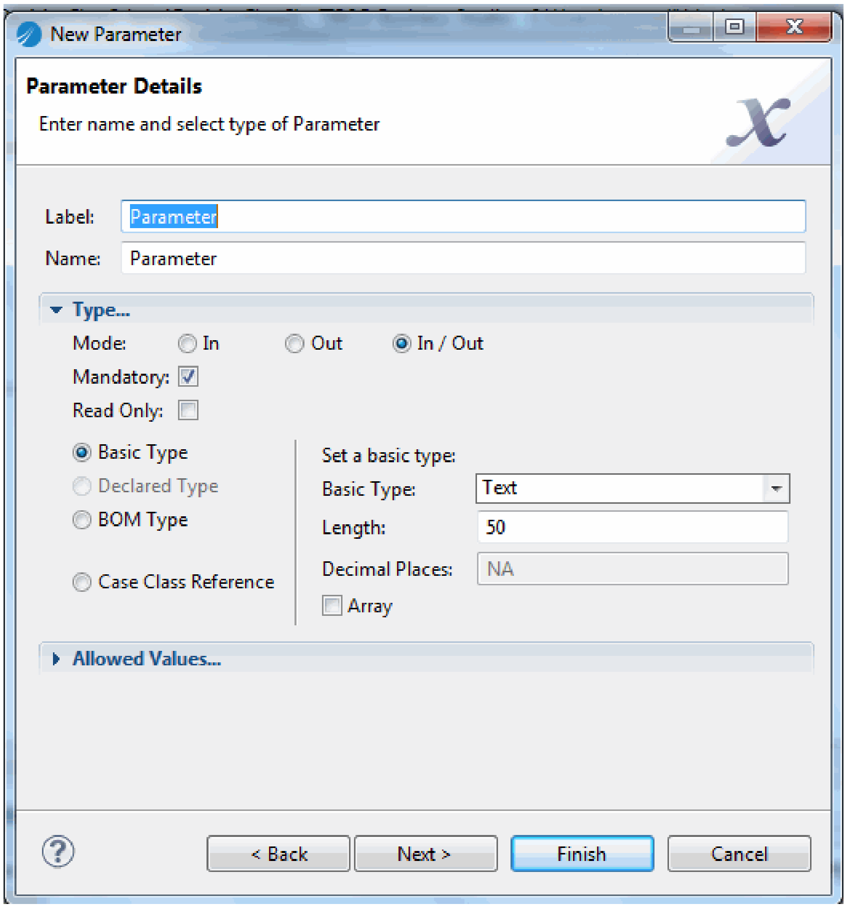
  For information about using the parameters in rules in the decisions table, see Adding Conditions and Actions to a Decisions Table on page 50.

Using the Wizard

To create a new parameter using the wizard, do the following:

1. In the Project Explorer, expand the decisions flow where you want to add a parameter.

2. In the Project Explorer, right-click the **Parameters** folder in the decisions flow, then select **New** > **Parameter**. The **New Parameter** dialog displays:



New Parameter

Parameter Details

Enter name and select type of Parameter

Label: Parameter

Name: Parameter

Type...

Mode: In / Out [✓ In / Out]

Mandatory: ✓

Read Only: ☐

[✓ Basic Type] Declared Type BOM Type Case Class Reference

Set a basic type:

Basic Type: Text

Length: 50

Decimal Places: NA

Array ☐

Allowed Values...

The selections on this dialog correspond to the columns in the **Parameters** tab when adding a parameter using the **Properties** view.

3. Complete the **New Parameter** dialog using the definitions provide in step 4 on page 39, then click **Finish**.

Chapter 4 Decisions Table Tasks

This chapter describes how to configure a decisions table task in a decisions flow.

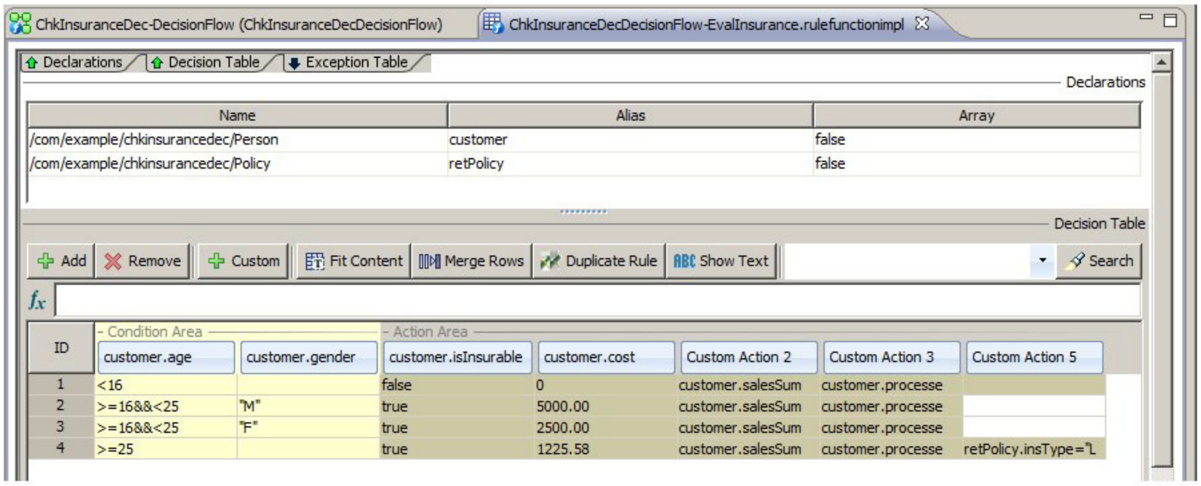
Topics

Decisions Table Task Editor

Each decisions table task on a decisions flow is configured separately. To view/configure a decisions table, either:

- Double-click on a decisions table task icon in a decisions flow, or
- select the decisions table task in the decisions flow, then click the **Edit Decision Table** button on the **General** tab of the **Properties** view.

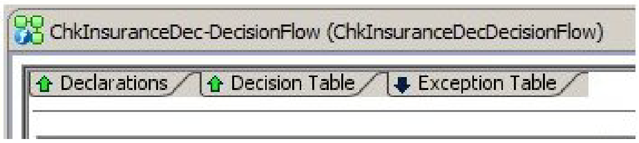
The decisions table editor is displayed:



Name	Alias	Array
/com/example/chkinsurancedec/Person	customer	false
/com/example/chkinsurancedec/Policy	retPolicy	false

	Condition Area		Action Area				
ID	customer.age	customer.gender	customer.isInsurable	customer.cost	Custom Action 2	Custom Action 3	Custom Action 5
1	<16		false	0	customer.salesSum	customer.processe	
2	>=16&&<25	"M"	true	5000.00	customer.salesSum	customer.processe	
3	>=16&&<25	"F"	true	2500.00	customer.salesSum	customer.processe	
4	>=25		true	1225.58	customer.salesSum	customer.processe	retPolicy.insType="L

The decisions table editor consists of three sections that can be displayed in the editor by clicking on the **Declarations**, **Decision Table**, and **Exception Table** tabs at the top of the editor. Any or all of the sections can be displayed at one time — a green up arrow indicates the section is currently displayed; a black down arrow indicates the section is not currently displayed:



These sections of the decisions table editor show the following:

- **Declarations** - Lists the parameters that can be used in the rules in the decisions table. Parameters are automatically added to this section when they are added to the decisions flow. For information, see Adding Parameters to a Decisions Flow on page 38.

- **Decision Table** - This section displays the decisions table, in which the decisions rules are defined.
- **Exception Table -** This is another instance of a decisions table where you can (optionally) enter conditions and actions for non-business logic. Each decisions table can optionally have an exception table. It enables you to separate the business logic of the main decisions table from any non-business logic. For instance, in the exception table, you can capture situations where fields are blank or contain invalid values, and define actions that send notifications or set return values. If you prefer, you can put non-business logic in the main table instead of using an exception table.

  The rules in this section are defined in the same way as the rules in the decisions table.

  The rules in the exception table are *always* evaluated/executed *after* the rules in the decisions table. There is also no method of setting rule execution priority in the exception table — the order in which the exception table rules are executed is indeterminate.

Defining Decisions Tables

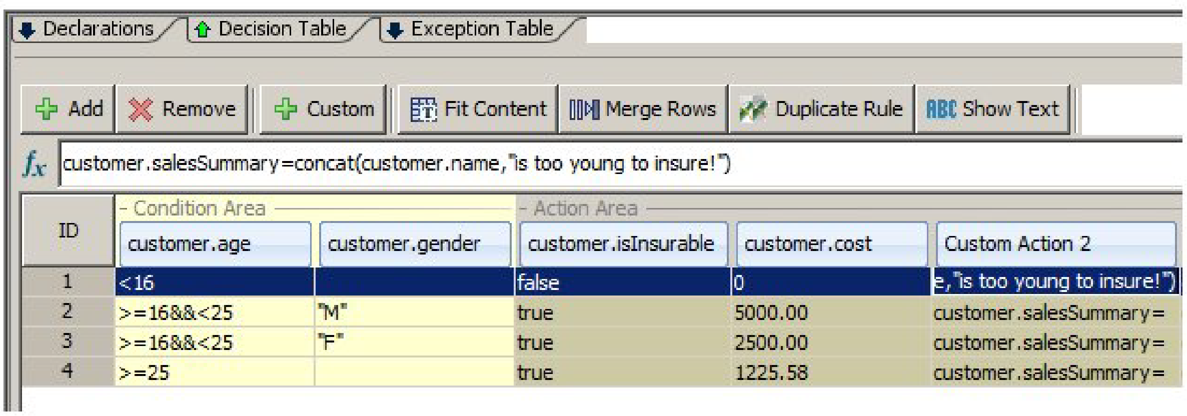
Decisions tables provide a method of building complex business rules. A decisions table rule is a row in the decisions table. It has one or more conditions and one or more actions. Each condition cell defines one condition, and each action cell defines one action.

Every condition must evaluate to true or false. If all conditions in a rule evaluate to true, the actions in the rule are executed.

You can add conditions and actions to a decisions table by dragging and dropping predefined parameters onto the table. The parameters reference the data objects from your business object model. You can then define threshold values (conditions) and actions in the cells of the table.

The conditions evaluate the data passed into the decisions service, and the actions determine the data returned from the decisions service.

The following is an example decisions table with conditions, actions, and rules defined:



fx customer.salesSummary=concat(customer.name,"is too young to insure!")

ID	Condition Area		Action Area		
	customer.age	customer.gender	customer.isInsurable	customer.cost	Custom Action 2
1	<16		false	0	e,"is too young to insure!")
2	>=16&&<25	"M"	true	5000.00	customer.salesSummary=
3	>=16&&<25	"F"	true	2500.00	customer.salesSummary=
4	>=25		true	1225.58	customer.salesSummary=

Each row in the decisions table constitutes a rule. In this example, there are four rules defined. Every rule in the table is evaluated when the decisions flow is executed (subject to possible priority settings — see Rule Execution Order/Priority on page 54).

The rule definitions consist of two areas:

- **Condition Area** - This is the area in which you will evaluate the attributes of parameters passed into the decisions service. One or more conditions can be added to this area. In the example above, there are two conditions in which the **age** and **gender** attributes of the **customer** parameter are evaluated. If there is no entry in a condition, for example for **customer.gender** in Rule 1, it evaluates to true by default.

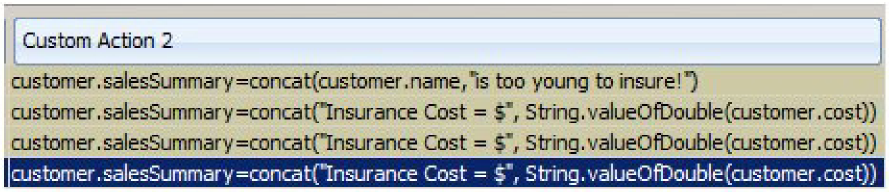
  Every condition must evaluate to true or false. If all conditions for a particular rule evaluate to true, the actions for that rule are executed.

- **Action Area** - This is the area in which you set the values of return data for the decisions service. For example, if the conditions for Rule 4 evaluate to true, the value of **customer.inInsurable** is set to true, the value of **customer.cost** is set to 1225.58, and so on for all of the defined actions for that rule.

For additional information about defining conditions and actions, see Defining Rules, page 49.

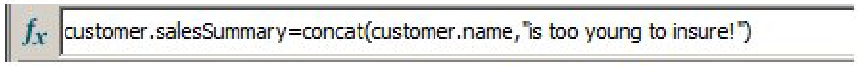
The decisions table provides buttons that allow you to do the following:

- **Add** - This is used to add a new rule to the table — see Adding Rules to a Decisions Table, page 53.
- **Remove** - This deletes the selected rule.
- **Custom** - This is used to add *custom* conditions and actions — Adding Custom Conditions and Actions, page 53.
- **Fit Content** - This expands the cells for a selected action or condition, if necessary, so that all of the action or condition is visible. For example:



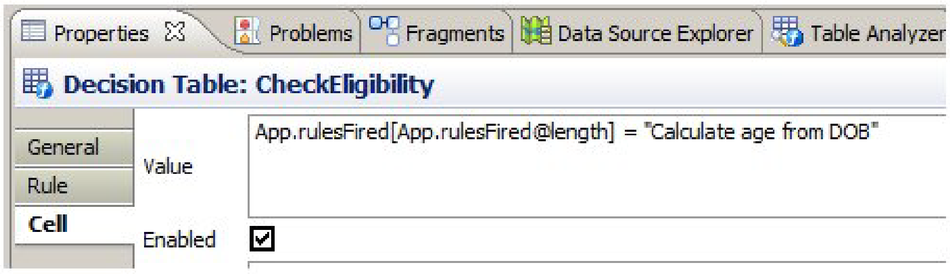
Custom Action 2
customer.salesSummary=concat(customer.name,"is too young to insure!")
customer.salesSummary=concat("Insurance Cost = $", String.valueOfDouble(customer.cost))
customer.salesSummary=concat("Insurance Cost = $", String.valueOfDouble(customer.cost))
customer.salesSummary=concat("Insurance Cost = $", String.valueOfDouble(customer.cost))

An alternative to expanding an entire column using this button is to select a particular cell. The entire contents of that cell is shown in the **fx** field directly below the decisions table buttons:



fx customer.salesSummary=concat(customer.name,"is too young to insure!")

You can also view the contents of a cell by clicking in the cell, then viewing the **Cell** tab in the **Properties** view:



Decision Table: CheckEligibility

Value: App.rulesFired[App.rulesFired@length] = "Calculate age from DOB"
Enabled: ✓

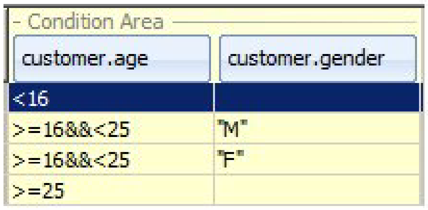
- **Merge Rows** - Merges rows with the same conditions into a single row. Actions for the first row are used for the merged row.

  To toggle between merged and unmerged rows, click the **Merge Rows** button again.

  The initial state of the merge rows button is determined by a decisions table preference. To set the initial state, select **Window > Preferences > Decisions > Decision Table** and set the property **Automatically merge rows** as desired.

- **Duplicate Rule** - Duplicates the currently selected row(s).

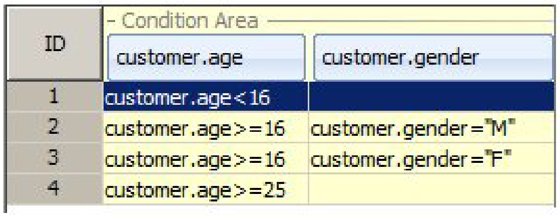
- **Show Text** - Displays the verbose contents of the table cells. For example, non-verbose displays this:



- Condition Area

customer.age	customer.gender
<16	
>=16&&<25	"M"
>=16&&<25	"F"
>=25	

Verbose displays this:



ID	customer.age	customer.gender
1	customer.age<16	
2	customer.age>=16	customer.gender="M"
3	customer.age>=16	customer.gender="F"
4	customer.age>=25	

Defining Rules

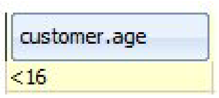
Defining a decisions table consists of defining rules, which consists of conditions and actions. These are explained in the following subsections.

Overview of Conditions and Actions

There are two types of conditions and actions that can be defined in a decisions table — regular and custom:

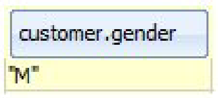
- **Regular Condition** - This type of condition evaluates the value of a single parameter attribute, and evaluates true or false. You can use condition operators, such as == (equal to), < (less than), > (greater than), >= (greater than or equal to), and so on (for a complete list of the valid operators, see Supported Operators on page 69).

  For example, this condition evaluates to true if the value of **customer.age** is less than 16:

  customer.age
  <16

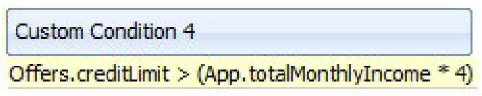
  In a regular condition, "equal to" is implied if a single value is provided in the condition cell. For example, this condition evaluates true if **customer.gender** is equal to "M":

  customer.gender
  "M"

  Note that an out-only (**Out** mode) parameter is null until you set it; don't check an out-only parameter in a condition until it is set.

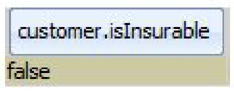
- **Custom Condition** - This type of condition evaluates the value of a parameter attribute using the available operators, standard functions, or script. These conditions allow you to use statements more complex than a comparison against a single parameter attribute. In these condition definitions, you use the *parameter.attribute* name in the statement. For example:

  Custom Condition 4
  Offers.creditLimit > (App.totalMonthlyIncome * 4)

All specified conditions must evaluate to true or false.

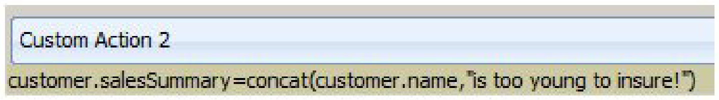
- **Regular Action** - This type of action sets the value of a single parameter attribute. In the following example, **customer.isInsurable** is set to false when this action is executed

  customer.isInsurable
  false

  The action specified is executed if all conditions in row (rule) evaluate to true, subject to specified rule execution order/priority — see Rule Execution Order/Priority, page 54.

- **Custom Action** - This type of action is used when you want to use the available operators, standard functions, or script. These types of actions allow you to use statements more complex than setting a single parameter attribute. For example:

  Custom Action 2
  customer.salesSummary=concat(customer.name,"is too young to insure!")

  The action specified is executed if all conditions in row (rule) evaluate to true, subject to specified rule execution order/priority — see Rule Execution Order/Priority, page 54.

For rules that are executed, the actions in the rule are executed in order, from left to right.

For a list of the operators that can be used in conditions and actions, see Supported Operators, page 69.

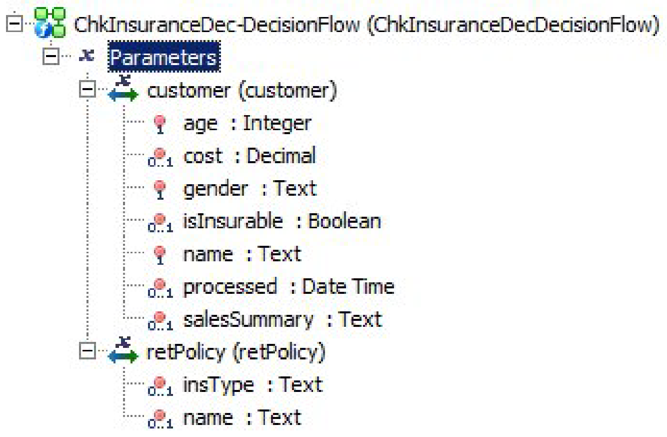
Note that when using the standard functions in custom conditions and actions, you can drag-and-drop the function into the action cell. For more information about using the standard functions, see Using the Standard Functions, page 65.

Adding Conditions and Actions to a Decisions Table

This section describes how to add:

- Regular conditions and actions — see Adding Regular Conditions and Actions, page 51
- Custom conditions and actions — see Adding Custom Conditions and Actions, page 53

To add either regular or custom conditions and actions, in Project Explorer, expand **Parameters** in the decisions flow in which you are adding conditions and actions:



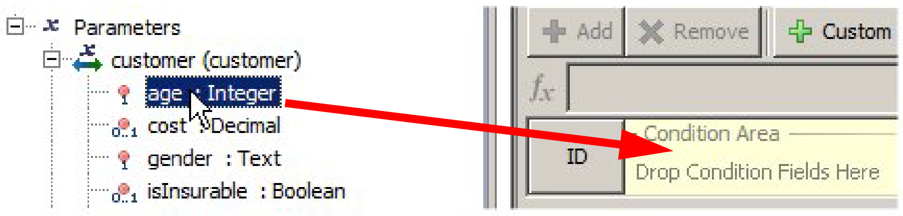
```
ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)
  Parameters
    customer (customer)
      age : Integer
      cost : Decimal
      gender : Text
      isInsurable : Boolean
      name : Text
      processed : Date Time
      salesSummary : Text
    retPolicy (retPolicy)
      insType : Text
      name : Text
```

You will need this expanded to add regular conditions and actions because you will drag-and-drop them onto the table; for custom conditions and actions, you will need to reference the attribute names for use in the statements of the custom conditions and actions.

Adding Regular Conditions and Actions

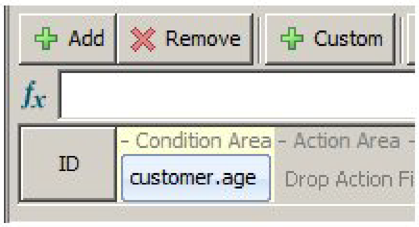
To add a regular condition or action to a decisions table:

1. To add a regular condition, drag the desired parameter from the Project Explorer, and drop it in the yellow Condition Area. For example, if you want to evaluate the age of a customer:

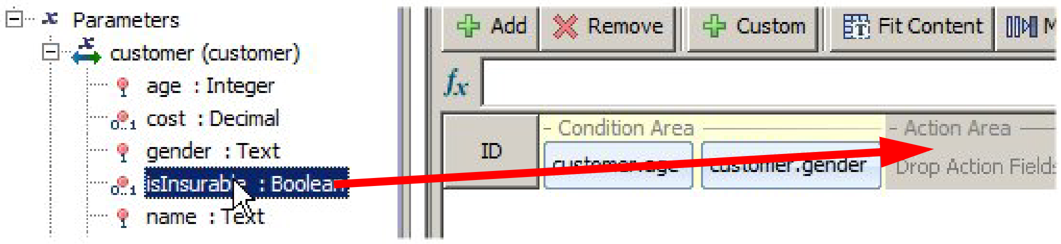


```
Parameters
  customer (customer)
    age : Integer
    cost : Decimal
    gender : Text
    isInsurable : Boolean
```

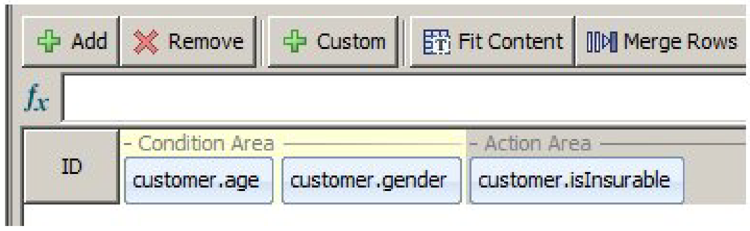
The parameter is now shown as a column header in the Condition Area:



2. Continue to drag and drop any additional attributes, into the Condition Area, that you are going to want to evaluate in the rules.
3. To add a regular action, drag the desired parameter from the Project Explorer, and drop it in the gray Action Area. For example, if you want to set the **isInsurable** flag, based on the conditions



The parameter is now shown as a column header in the Action Area:

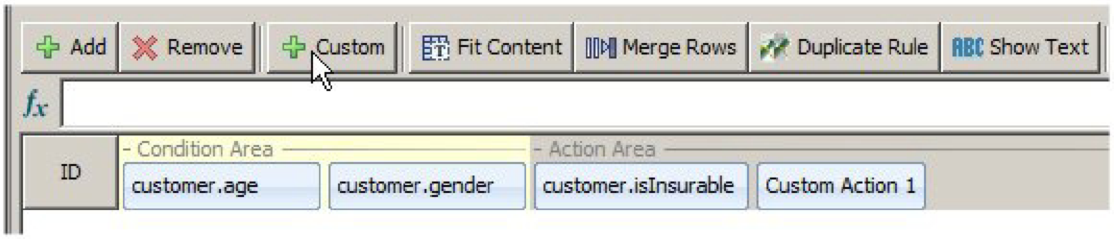


4. Continue to drag and drop any additional attributes, into the Action Area, that you want to set when all conditions in the rule evaluate to true.
5. You can now define the rules for the added conditions and actions. — see Adding Rules to a Decisions Table, page 53.

Adding Custom Conditions and Actions

1. To add a custom condition or action, click on the **Custom** button in the decisions table editor and select **Add a Custom Condition** or **Add a Custom Action** from the drop-down menu, as appropriate.

   A new column header is added to the Condition Area or Action Area, depending on which you selected:



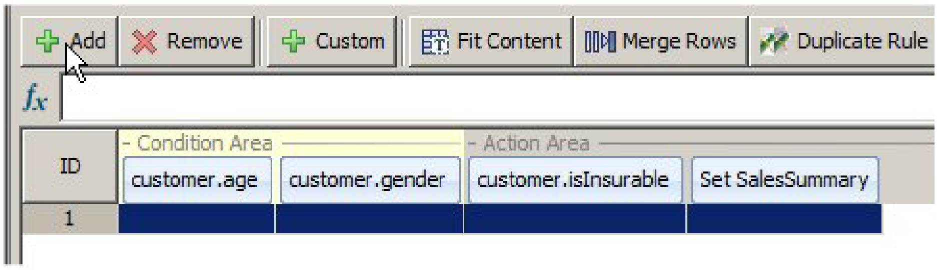
2. Optionally, right click on the newly added custom condition or action column header, and select **Field Settings**. This displays a dialog that allows you to customize the name of the column heading.
3. Continue to add additional custom conditions and actions as necessary.
4. You can now define the rules for the added conditions and actions — see Adding Rules to a Decisions Table, page 53.

Adding Rules to a Decisions Table

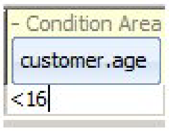
After you've added conditions and actions to your decisions table, you can add the rules that evaluate the conditions and set attribute values in the actions (note that you could actually define the rules immediately after adding the first condition to the table).

To add a rule:

1. Click the **Add** button in the decisions table editor. A row is added to the table:

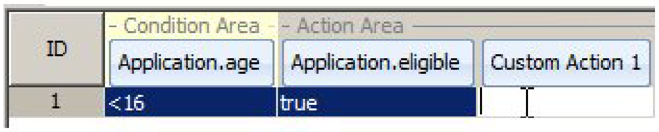


2. Click in the cells of the rule and add the desired conditions and actions. For example:



- Condition Area
customer.age
<16

You can drag and drop parameter attributes into custom condition and action cells. Note, however, you must first click in the cell into which you want to add the attribute, then drag it from Project Explorer and drop it in the cell (the cell must have a white background before you attempt to drag to it):



ID	- Condition Area	- Action Area	
	Application.age	Application.eligible	Custom Action 1
1	<16	true	

For additional information about defining the conditions and actions in the rule, see Defining Rules, page 49.

3. Continue to add additional rules as needed by clicking the **Add** button.

Working With Conditions, Actions, and Rules

This section describes various features of working with conditions, actions, and rules that have been added to a decisions table.

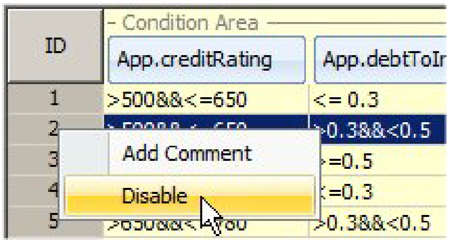
Rule Execution Order/Priority

By default, all of the rules in a decisions table have the same priority (5). When the rules all have the same priority, the order in which the rules are evaluated is indeterminate. The runtime engine might evaluate row 2 first, then row 5, then row 1, and so on. In this scenario (equal priority), all rules whose conditions all evaluate to true, are executed, meaning its actions are executed (for rules that are executed, the actions in the rule are executed in order, from left to right).

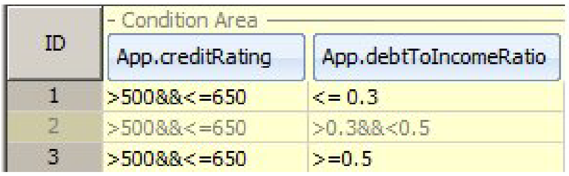
Note, however, you can configure rules, as well as cells, so that they are not executed (disabled), they are executed in a certain order (priority), or that only a single rule be executed. These are described in the following subsections.

Disabling Rules and Cells

To disable a rule so that it is not evaluated or executed, right-click on the rule ID, then select **disable**:



The entire rule becomes grayed out, indicating it is currently disabled:

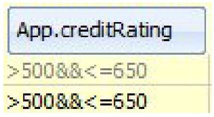


Condition Area

ID	App.creditRating	App.debtToIncomeRatio
1	>500&&<=650	<= 0.3
2	>500&&<=650	>0.3&&<0.5
3	>500&&<=650	>=0.5

To re-enable a rule, right-click on the rule ID, then select **Enable**.

To disable an individual cell so that that cell is not evaluated (if it's a condition) or executed (if it's an action), right-click on the cell, then select **Disable**. The individual cell becomes grayed out, indicating it is currently disabled:



App.creditRating
>500&&<=650
>500&&<=650

You can also disable an individual cell by clicking in a cell, displaying the **Cell** tab in the **Properties** view, then unchecking the **Enabled** box.

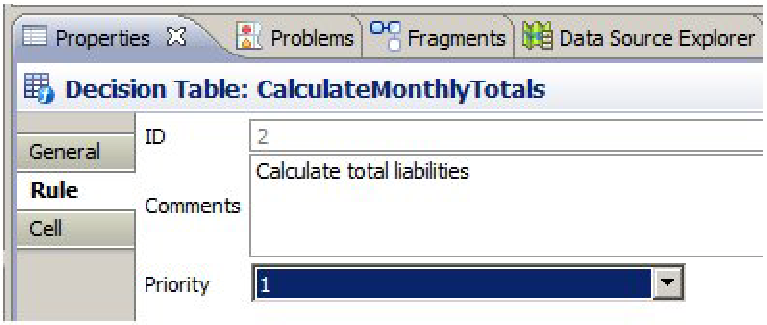
To re-enable a cell, right-click on the cell, then select **Enable**, or check the **Enabled** box on the **Cell** tab.

Setting Rule Priority

To change the priority for a rule:

1. Select the desired rule by clicking in the rule ID.
2. Display the **Rule** tab in the **Properties** view.

3. Select the desired priority value from the **Priority** field drop-down list
   - 1 is the highest priority
   - 10 is the lowest priority



Decision Table: CalculateMonthlyTotals

ID: 2

Comments: Calculate total liabilities

Priority: 1

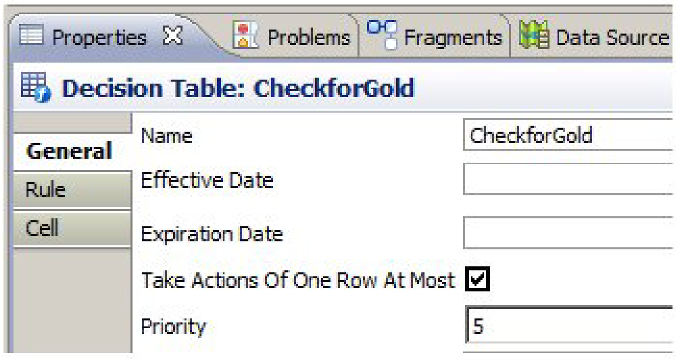
When changing the priority, ensure the ID number shown in the **ID** field on the **Cell** tab is the correct one for the rule you want to change.

Rows with higher priorities are executed before those with lower priorities, as follows:

1. First, all conditions are checked for all rules that have the highest priority. (The checking order within a set of rules with the same priority is indeterminate.)
2. Then the actions for all of those rules whose conditions evaluate to true are executed. (The execution order is indeterminate. The runtime engine optimizes rule execution.)
3. The process is repeated for all rules with the next highest priority, and so on.

Executing a Single Rule

You can specify that only the *first rule* whose conditions all evaluate to true be executed. To specify this, check the **Take Actions Of One Row At Most** box on the decisions table's **General** tab in the **Properties** view:



Decision Table: CheckforGold

Name: CheckforGold
Effective Date:
Expiration Date:
Take Actions Of One Row At Most: ☑
Priority: 5

The rules are still evaluated according to any priorities specified, but only the first one is executed.

The Effective Date and Expiration Date features are not supported with decisions tables.

Empty Cells

Leaving a condition cell in a decisions table empty causes that condition to always evaluate true. This can be done for both regular and custom conditions.

You can also include an asterisk in a regular condition cell to cause it to evaluate to true; do not use an asterisk in a custom condition cell for this purpose.

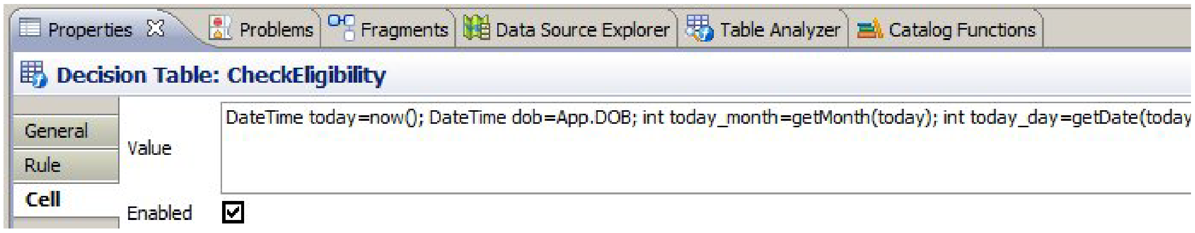
And you can include **true** in a custom condition cell to cause it to evaluate to true; do not use this in regular condition cells, however.

Of these methods, the most efficient is to leave the cell empty. This results in the condition not being evaluated.

Also see Disabling Rules and Cells, page 55.

Using Script in a Custom Condition or Action

The *ActiveMatrix decisions rules language* (which is a JavaScript-like language) can be used in custom conditions and actions. It must be typed directly into the appropriate cell, and it can only be entered and viewed as a single line. For script that is lengthy, it may be best to view it by clicking in the cell in which script has been entered, then view the **Cell** tab on the **Properties** view. For example:



Value: DateTime today=now(); DateTime dob=App.DOB; int today_month=getMonth(today); int today_day=getDate(today

Enabled: ✓

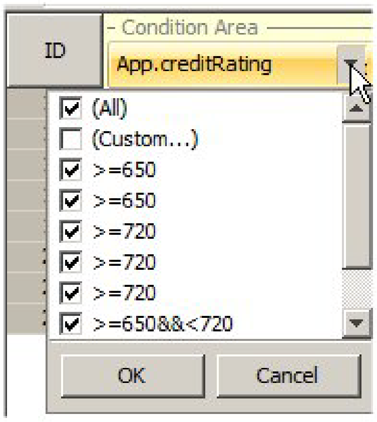
For information about the ActiveMatrix decisions rules language, see ActiveMatrix Decisions Rules Language on page 119.

Filtering Rules

You can filter rules in the decisions table so that only the desired rules are displayed based on filter criteria you enter in one or more condition or action columns.

The filter feature must be enabled on the **Preferences** dialog to be able to filter rules. To enable it, select **Window > Preferences > Decisions** > **Decision Table**, then check the **Show column filter** box.

When the rules filter is enabled, a drop-down list is available for each condition and action column in the table:



To Filter Rows

1. Click the down arrow to the right of a column header. You see a list of all values that have been used in the column's cells.
2. Do any of the following, as appropriate:
   - Select the **All** check box to uncheck all the other check boxes, then select the check boxes beside the value or values you are interested in.
   - Unselect the check box for values you are not interested in.
3. Click **OK**. Only rows containing the checked values display.
4. Repeat to filter on additional columns as desired.
5. To remove the filter conditions when you are done, select **All** in each of the filter columns.

Removing Rules, Conditions, and Actions

To delete a rule from a decisions table, select the desired row, then click the **Remove** button.

To delete a condition or action, right-click on the column header for the desired condition or action, then select **Remove**.

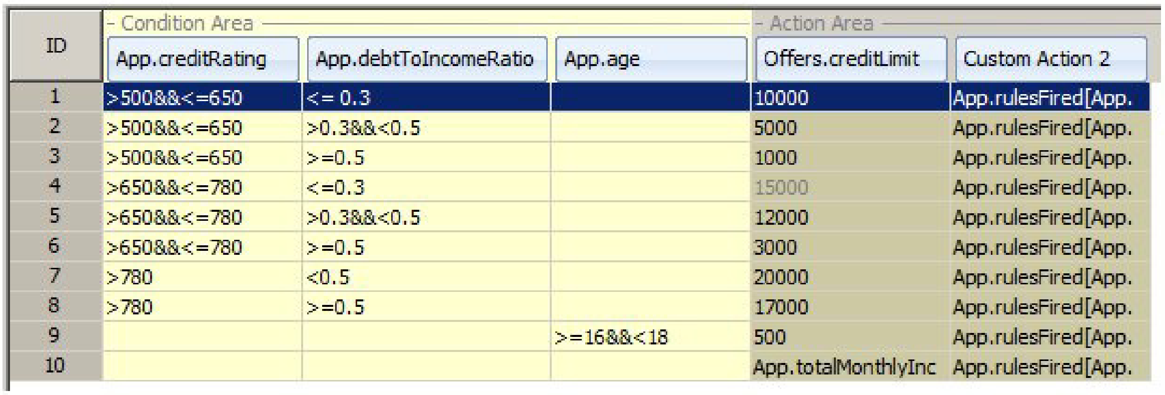
Moving Conditions or Actions

You can rearrange the order of columns within the Condition Area and the Action Area. To rearrange columns, right-click the column you want to move and select **Move**. A sub-menu is displayed with the following options:

- Move to the beginning
- Move to the left
- Move to the right
- Move to the end

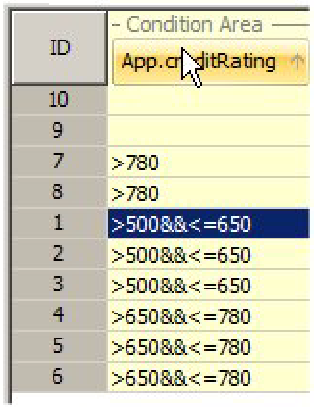
Sorting Rules

By default, rules are shown in the table in the order in which they were entered, with the **ID** column showing this order:



ID	Condition Area			Action Area	
	App.creditRating	App.debtToIncomeRatio	App.age	Offers.creditLimit	Custom Action 2
1	>500&&<=650	<= 0.3		10000	App.rulesFired[App.
2	>500&&<=650	>0.3&&<0.5		5000	App.rulesFired[App.
3	>500&&<=650	>=0.5		1000	App.rulesFired[App.
4	>650&&<=780	<=0.3		15000	App.rulesFired[App.
5	>650&&<=780	>0.3&&<0.5		12000	App.rulesFired[App.
6	>650&&<=780	>=0.5		3000	App.rulesFired[App.
7	>780	<0.5		20000	App.rulesFired[App.
8	>780	>=0.5		17000	App.rulesFired[App.
9			>=16&&<18	500	App.rulesFired[App.
10				App.totalMonthlyInc	App.rulesFired[App.

You can sort the table of rules by clicking a column header. This causes an "up arrow" to appear in the column header, indicating that the table is now sorted in ascending alphanumeric order by the values in that particular column:



ID	Condition Area
	App.creditRating
10	
9	
7	>780
8	>780
1	>500&&<=650
2	>500&&<=650
3	>500&&<=650
4	>650&&<=780
5	>650&&<=780
6	>650&&<=780

Note that the empty cells are always first when sorting in ascending order.

Clicking the same column header again causes the rules to be sorted in descending order — the arrow changes to a "down arrow". Clicking the column again (when the down arrow is shown) causes the table to go back to the default order.

Searching

You can search for a particular value in the decisions table by entering a value in the field to the left of the **Search** button, then clicking on the **Search** button.

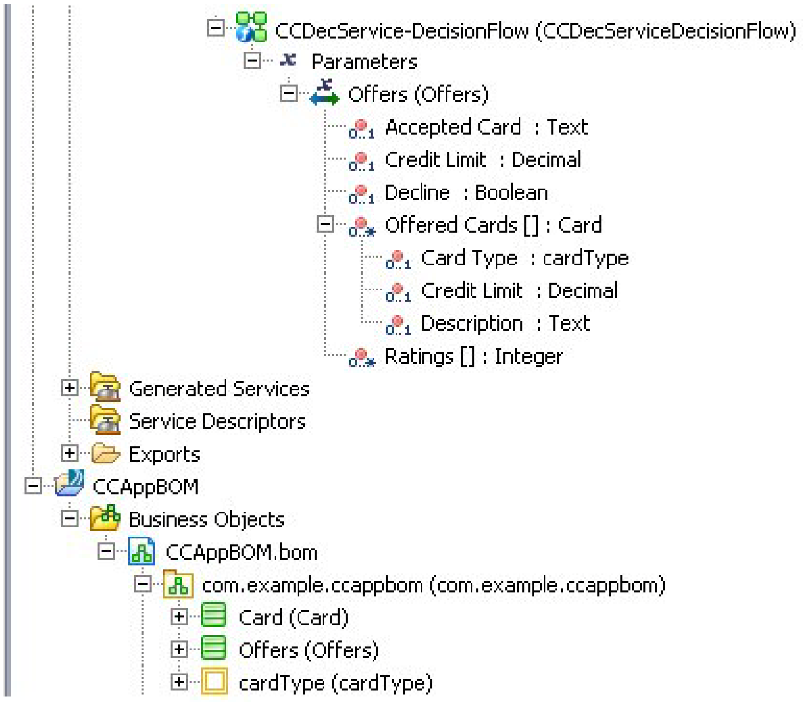
The next rule (from the one that is currently selected (highlighted)) that contains the specified value is selected. Each successive click of the **Search** button finds the next occurrence of the value.

Using Arrays in Conditions and Actions

This section explains how to create conditions and actions that use arrays.

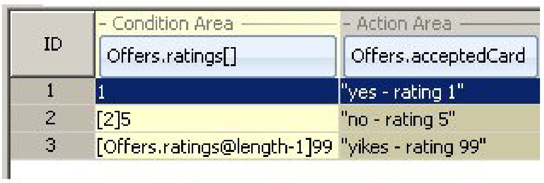
Arrays of BOM types cannot be used in actions or conditions. However, BOM types can have attributes that are arrays, including nested (that is, contained or referenced) BOM arrays, as well as arrays of primitive types. For example, if the BOM contains a class type of Account and Person, an array of Person objects cannot be used in a condition or action. If Person has an attribute of accounts (0..*), then Person.accounts[] could be used in a condition or action.

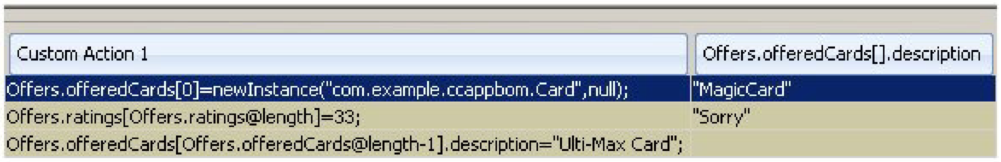
The following is an example showing some BOM types and formal parameters that have been defined in a decisions flow:



```
CCDecService-DecisionFlow (CCDecServiceDecisionFlow)
  Parameters
    Offers (Offers)
      Accepted Card  : Text
      Credit Limit  : Decimal
      Decline  : Boolean
      Offered Cards [] : Card
        Card Type  : cardType
        Credit Limit  : Decimal
        Description  : Text
      Ratings [] : Integer
Generated Services
Service Descriptors
Exports
CCAppBOM
  Business Objects
    CCAppBOM.bom
      com.example.ccappbom (com.example.ccappbom)
        Card (Card)
        Offers (Offers)
        cardType (cardType)
```

And the following is an example decisions table using arrays from the example above (the table is split in the illustration for legibility):





ID	Condition Area	Action Area		
	Offers.ratings[]	Offers.acceptedCard	Custom Action 1	Offers.offeredCards[].description
1	1	"yes - rating 1"	Offers.offeredCards[0]=newInstance("com.example.ccappbom.Card",null);	"MagicCard"
2	[2]5	"no - rating 5"	Offers.ratings[Offers.ratings@length]=33;	"Sorry"
3	[Offers.ratings@length-1]99	"yikes - rating 99"	Offers.offeredCards[Offers.offeredCards@length-1].description="Ulti-Max Card";	

To use a primitive array attribute, drag and drop the array (for example, Offers.ratings[] is an array of integers). For an attribute that is an array of business objects, drag and drop the desired attribute. For example, offeredCards[] is an array of business objects of type Card, and description is an attribute of Card that the action is going to set.

Specifying Values in Arrays

When the expression in the column header contains the access to a position within an array (for example, Offers.offeredCards[].description in the example above), then you can specify which position to use in the cell of a table row. To specify the position, enclose the position index within square brackets, followed by the value. For example, in the condition column for Offers.ratings[], to check if the third element of the array is equal to 5, you would enter the following:

```
[2]5
```

For an action, this would assign 5 to the third element.

Indexes start with 0.

Omitting the Array Index

If you omit the array index, and simply set the cell to a value, like row 1 in the illustration above, by default, the array component's last element is used. So if the Offers.ratings[] array had the following elements:

5
10
15
23

Then in the condition column for row 1, where the cell value is 1, the condition would not be met since the last entry in the array is 23. Similarly, in the action column, “MagicCard” would be assigned to the description on the last Card in the Offers.offeredCards[] array.

- To avoid ambiguity, you must specify the array index in the following cases:
  - Multiple loop indexes
  - String expressions
- If you omit the array index, and no value is present at the last element, an exception results. It is recommended that you specify the index, and ensure that all indexes specified have values. Use the **newInstance** function as shown above to create business objects for array entries. (Note that the second argument to the **newInstance** function is not used and can be null). You must specify the fully qualified name of the business object (such as com.example.ccappbom.Card; using only Card will result in an exception). For primitive types, you can append to the end of the array by doing the following:

  ```
  Offers.ratings[Offers.ratings@length]=33;
  ```

Using Function Evaluations

You can also specify index values using a function evaluation, for example:

```
[Number.intValue("0",10)]10
```

Using Multiple Loop Indexes

If multiple loop indexes are involved, as in the case of nested business objects, use the colon character (:) as the delimiter for indexes. Always specify the index, to avoid ambiguity. For example, given this nested object situation:

```
a.b[].c[].d[].intProp
```

You might specify a condition cell value as follows:

```
[0:1:System.getGlobalVariableAsInt(%%GV%%)]>=100
```

Which is the equivalent of the following:

```
a.b[0].c[1].d[System.getGlobalVariableAsInt(%%GV%%)] >= 100
```

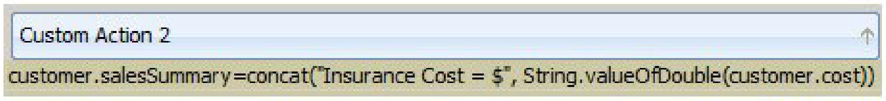
For more examples of using arrays, see the "Sample-Array" and "Sample-NewInstance" projects provided in the samples that are included when you install the TIBCO Business Studio ActiveMatrix Decisions Add-in. These are included in the **ADEC_Samples.zip** file, which is located in the directory, *STUDIO_HOME***studio****3.***n***sda****samples**\.

Validating Rules in a Decisions Table

After you've finished adding conditions, actions, and rules to your decisions table, you can validate the table by using the Table Analyzer. For information about the analyzer, see Validating Decisions Tables, page 71.

Using the Standard Functions

A catalog of standard functions is provided that can be used in custom conditions and actions. For example, the following custom action uses the **concat** and **valueOfDouble** functions:



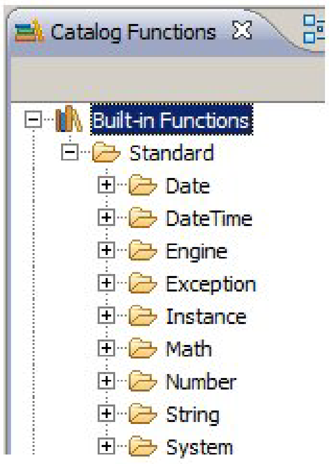
Custom Action 2

```
customer.salesSummary=concat("Insurance Cost = $", String.valueOfDouble(customer.cost))
```

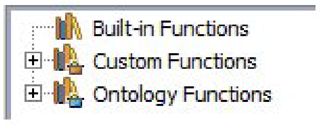
To view the available standard catalog functions in TIBCO Business Studio, select **Window** > **Show View** > **Other** > **ActiveMatrix Decisions** > **Catalog Functions**.

A decisions table needs to be opened before you can view the standard functions as described here.

The **Catalog Functions** tab is displayed. Drill down to the standard functions:



Also note that there is an anomaly that may prevent the standard functions from displaying properly when you first open the editor:



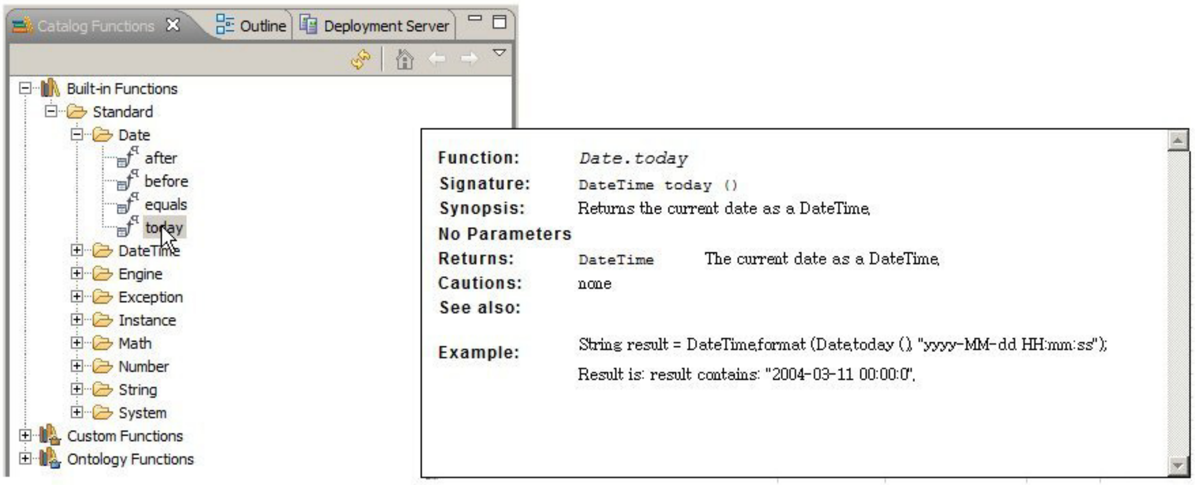
To work around this issue, click on any editor tab. The standard functions will then display properly.

The standard functions provide the following types of functionality:

- **Date** functions allow you to compare two DateTime values using only the date portion of the value.
- **DateTime** functions allow you to perform these date/time related tasks and more: add units of time to a DateTime, compare, retrieve, and format dates and times. (Also see Dates and Times on page 28 for additional information about using dates and times in decisions table rules.)
- **Engine** functions allow you to retrieve information about the TIBCO Business Studio ActiveMatrix Decisions Add-in, for example, the number of decisions services fired.
- The **Exception** function enables you to create an exception.
- **Instance** functions allow you to create and delete parameter object instances and perform other instance-related tasks, for example, return an instance given an internal ID.
- **Math** functions allow you to perform advanced mathematical operations.
- **Number** functions allow you to perform type conversions from and to numbers and return the maximum and minimum values for a numeric type.
- **String** functions allow you to perform comparisons, searches, conversions, and other operations with strings.
- **System** functions allow you to send messages to a debug log, retrieve global variables, retrieve system properties, and write data to a file.

Viewing Function Descriptions and Syntax Information

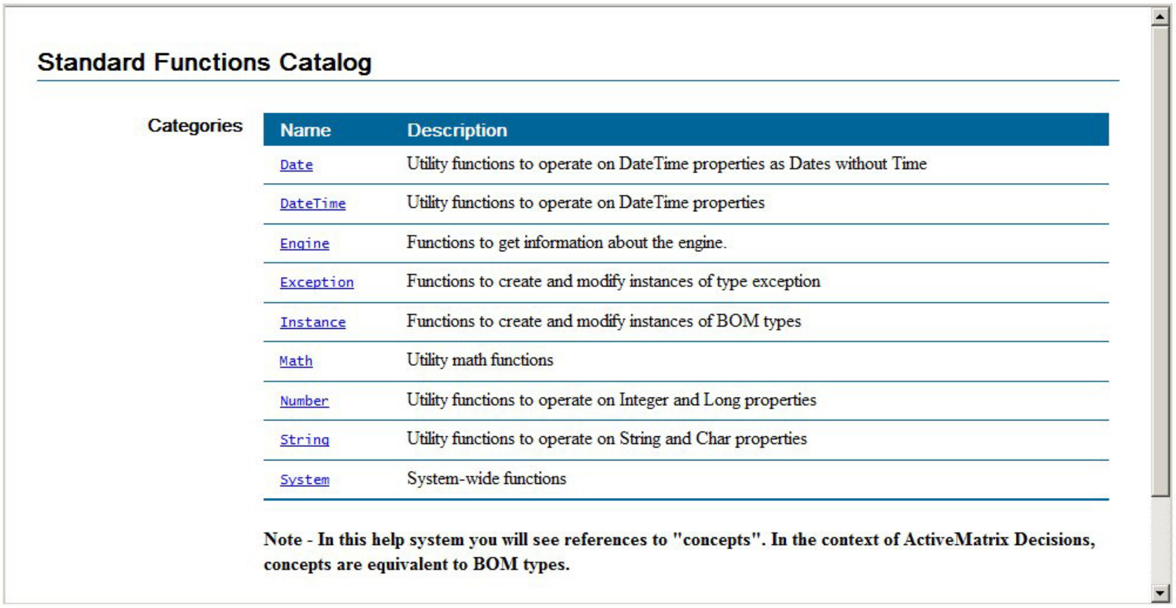
You can view a description of each standard function, as well as syntax details, by hovering the mouse pointer over any one of the functions in the catalog. For example:



Function: *Date.today*
Signature: `DateTime today ()`
Synopsis: Returns the current date as a DateTime.
No Parameters
Returns: `DateTime` The current date as a DateTime.
Cautions: none
See also:
Example: String result = DateTime.format (Date.today (), "yyyy-MM-dd HH:mm:ss");
Result is: result contains: "2004-03-11 00:00:0".

There is also an HTML-based help system available that provides descriptions and syntax information for each of the standard functions. This help system is invoked by executing the following:

*STUDIO_HOME***studio\3.***n***\sda\doc\functions\index.html**



Standard Functions Catalog

Categories

Name	Description
Date	Utility functions to operate on DateTime properties as Dates without Time
DateTime	Utility functions to operate on DateTime properties
Engine	Functions to get information about the engine.
Exception	Functions to create and modify instances of type exception
Instance	Functions to create and modify instances of BOM types
Math	Utility math functions
Number	Utility functions to operate on Integer and Long properties
String	Utility functions to operate on String and Char properties
System	System-wide functions

Note - In this help system you will see references to "concepts". In the context of ActiveMatrix Decisions, concepts are equivalent to BOM types.

Adding a Standard Function to a Condition or Action

You can drag-and-drop functions from the catalog into a condition or action cell, then type in any other necessary values or operators for the condition or action statement.

Supported Operators

The following table lists the operators supported for use in the conditions and actions in decisions tables:

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
||	Logical OR
&&	Logical AND
+	Addition or String concatenation (depending on context)
!	Logical NOT (only with boolean)
-	Subtraction
*	"Don't Care" (that is, ignore), or multiplication (depending on context)
/	Division
=	Assignment
.	Scope resolution
()	Operator precedence order or function call (depending on context)
[]	Array declaration or array indexing (depending on context)
{}	Block resolution or array Initialization (depending on context)
?:	Ternary conditional
++	Increment

Operator	Description
--	Decrement

Chapter 5 Validating Decisions Tables

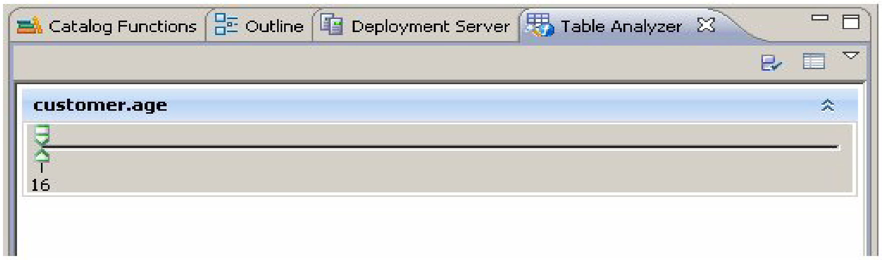
The Table Analyzer provides various tools that you can use to check the validity of your decisions tables.

Topics

Overview of Table Analyzer

The Table Analyzer provides various tools that you can use to check the validity of your decisions tables.

To view the Table Analyzer in TIBCO Business Studio, select **Window** > **Show View** > **Other** > **ActiveMatrix Decisions** > **Table Analyzer**. The **Table Analyzer** view is displayed:



The Table Analyzer allows you to set example condition values and perform various validation checks on the currently displayed decisions table.

Table analyzer supports the Not Equal To (!=) operator.

When you open a table, the Table Analyzer view dynamically creates controls for setting values and ranges for each condition in the table.

If the condition is a range, for instance "< 40" or "> 10 && < 100", then the corresponding control in the Table Analyzer view is a Slider that spans the range from the minimum value to the maximum value. You can then select a range of values within the range. The arrow above shows the maximum value and the arrow below shows the minimum value in the range.

Analyze Table

The **Analyze Table** button does not use the information in the Table Analyzer view. It analyzes the decisions table directly for the following potential issues.

Ranges

Uncovered ranges Finds gaps in coverage for the selected column's conditions. Suppose a condition column specifies ranges of values for the business object attribute client_age. The table has one row that specifies the range: > 18 && < 25 and another that specifies the range > 30 && < 100. Table Analyzer would report that the following values will not trigger an action: Values between 25 and 30; values less than 18; and values greater than 100

Overlapping ranges Using the client age range example again, suppose you set one condition as > 18 && < 25 and another as > 21 && < 100. Table Analyzer would report that these two ranges overlap.

Conflicting Actions If two rows have identical sets of conditions but different actions, it is detected as a potential issue by the analyzer.

How Date Ranges Appear Dates are treated as numeric ranges. For example the range 20080612 to 20090612 represents the data range for June 12, 2008 to June 12, 2009.

Rules that can be combined

If two rows have different conditions but the same actions, Table Analyzer reports that you can combine the conditions. For example, suppose one condition specifies > 50, and another specifies < 30, and they both have the same actions. Table Analyzer would report that you can combine these conditions as: > 50 || < 30. (greater than fifty or less than thirty.) In this case, you would remove one rule and update the other one accordingly.

Show Coverage

When you click the **Show Coverage** button, Table Analyzer highlights all rows in the table that meet all the criteria you specify in the Table Analyzer view.

For example, if you select Male for Gender; true for Eligible; and 50000-60000 for Income, clicking Show Coverage displays all rows that have Male, true, and an income between 50000 and 60000.

If a certain value *x* is selected for Show Coverage, it will not match any conditions which have value !=x.

Options for Range Conditions

For conditions that specify ranges, you can select either of the following behaviors, using an option in Preferences. See Setting Table Analyzer Preferences on page 75.

- Table Analyzer highlights rows where all of the values specified in the conditions fall within the specified ranges. (This is the default behavior.) For example, if the slider in the Table Analyzer view is set for a range between 55 and 60, Decision Manager highlights a row whose condition specifies > 58 && < 60. But it does not highlight a row whose condition specifies > 50 && < 70.

- Table Analyzer highlights rows where some of the values specified in the conditions fall within the specified ranges. In this case, Table Analyzer would display the row from the example above, where the condition specifies > 50 && < 70.

Setting Table Analyzer Preferences

You can set preferences for table range conditions and to determine if table analyzer appears by default when you display a decisions table.

1. From the menu select **Window** > **Preferences**.
2. In the **Preferences** dialog, select **Decisions** > **Decision Table** > **Analyzer**.
3. Set your preferences (see below).
4. Click **Apply** if you want to set preferences in another dialog before committing all the preferences.
5. Click **OK** to save and close the preferences editor.

Initially Show Table Analyzer View Option

- To show Table Analyzer view whenever you display a decisions table, check the Initially show Table Analyzer view check box.
- To hide Table analyzer view when you display a decisions table, uncheck the Initially show Table Analyzer view check box.
- To highlight rules in a decisions table, which can be triggered on selecting the values in the analyzer view.

Highlight Partial Ranges

- To highlight only rows whose condition falls completely within the specified condition range, uncheck the Highlight Partial Ranges check box.
- To highlight all rows whose condition fall partially within the specified condition range, check the Highlight Partial Ranges check box.

Use Domain Model for Table Completeness

- To report any uncovered domain entries.

Analyze while opening table

- If there is any equal set of conditions but with different set of actions, the analyzer will report it.
- If multiple rules can be combined into one, the analyzer will report them.

Validating Decisions Tables

TIBCO Business Studio ActiveMatrix Decisions Add-in validates the decisions tables on startup, and every time you save a table. On startup, all existing tables are validated for syntax.

If there are any access control violations or syntax errors in the tables, they are shown in the Problems View tab. New errors in syntax are added to these existing errors. Double-click errors to see the problematic view. Take any needed corrective actions and then validate the table again until all errors are resolved.

NullPointerExceptions are silently ignored. They occur when a null String is passed to a function that does not check for null, or because you accessed a property of a null contained concept (*parent.child.property* where *child* is null).

If a condition table cell is empty, contains an asterisk, or is disabled, it is skipped and treated as if it evaluates to true.

Chapter 6 Calling a Decisions Service from a BPM Process

This chapter describes how to call a decisions service from a BPM process.

Topics

Introduction

There are a number of ways in which you can call a decisions service from a BPM process:

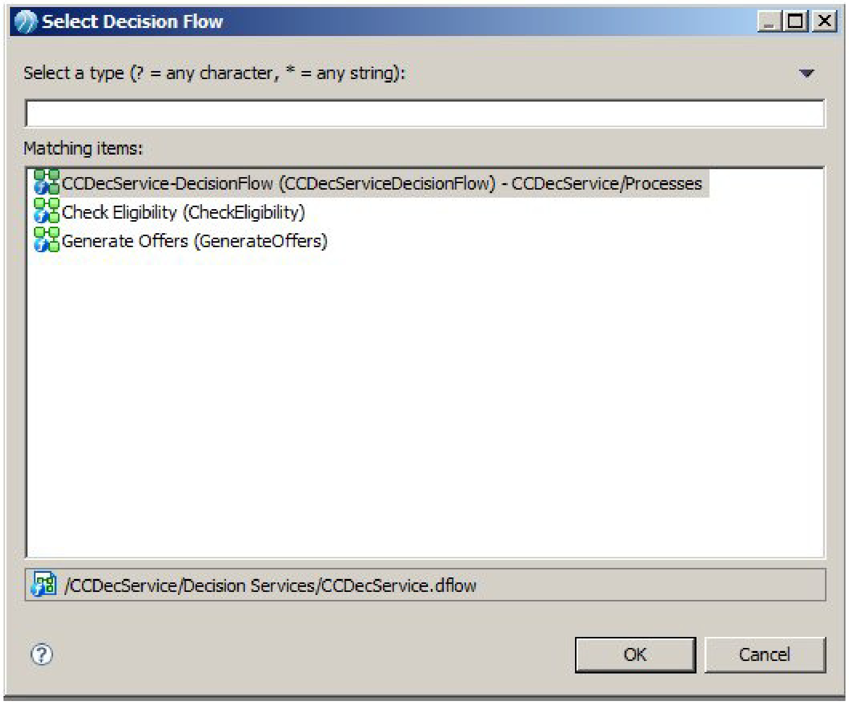
- **Add a decisions service task to a BPM process** — see Adding and Configuring Decisions Service Task on page 79
- **Drag and drop a decisions flow onto a BPM process** — see Drag-and-Drop a Decisions Flow onto a BPM Process on page 82
- **Generate a decisions flow from a BPM process** — see Generate a Decisions Flow from a BPM Process on page 84

Adding and Configuring Decisions Service Task

To call a decisions service from a BPM process, you can add a *decisions service task* to your process. In the runtime environment, the decisions service is executed when the decisions service task is reached in the process flow. To add and configure a decisions service task, do the following:

Task A Add a Decisions Service Task to Your Business Process

1. From the Task Palette, drag a service task to your BPM process.
2. On the **General** tab for the service task, select **Decision Service** from the **Service Type** field drop-down list.
3. Click on the **Open Decision-Flow** link in the **Invocation of Decision-Flow** field. The Select Decision Flow dialog displays.


Select Decision Flow

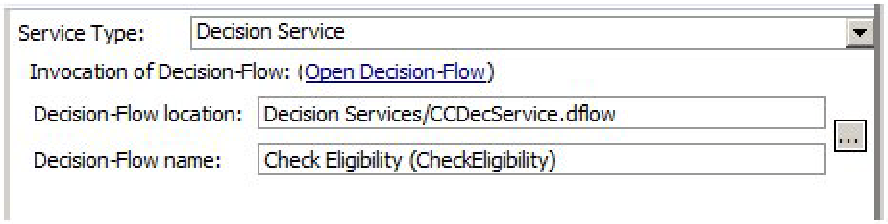
Select a type (? = any character, * = any string):

Matching items:

CCDecService-DecisionFlow (CCDecServiceDecisionFlow) - CCDecService/Processes
Check Eligibility (CheckEligibility)
Generate Offers (GenerateOffers)

/CCDecService/Decision Services/CCDecService.dflow

4. Select the decisions flow you want to invoke and click **OK**. The decisions service details are automatically completed in the **General** tab of the **Properties** view.



Service Type: Decision Service

Invocation of Decision-Flow: (Open Decision-Flow)

Decision-Flow location: Decision Services/CCDecService.dflow

Decision-Flow name: Check Eligibility (CheckEligibility)

— To view the decisions flow in the decisions flow editor, click **Open Decision-Flow**.

— To choose a different decisions flow, click the browse button to the right of the **Decision Flow location** field. The **Select Decision Flow** dialog displays. Select a different decisions flow and click **OK**.

Task B Define Parameters in the BPM Process

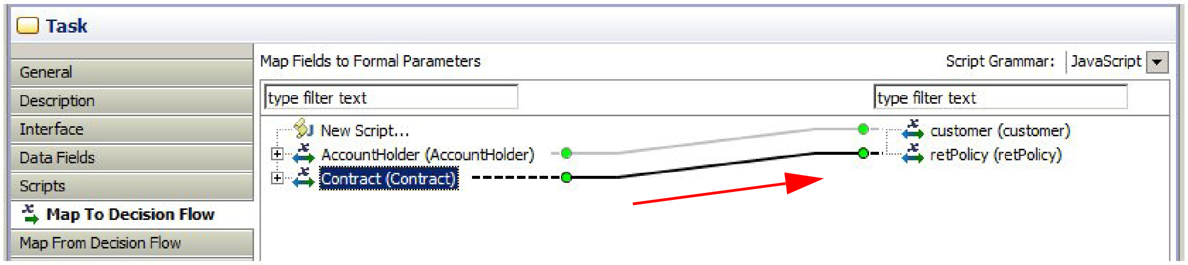
If parameters had not already been defined in the BPM process, you must do so before mapping parameters to and from the decisions service task. This is done in the same way as parameters are added to a decisions flow. For information, see Adding Parameters to a Decisions Flow on page 38. Or refer to the TIBCO Business Studio documentation.

Task C Map Parameters To and From the Decisions Service

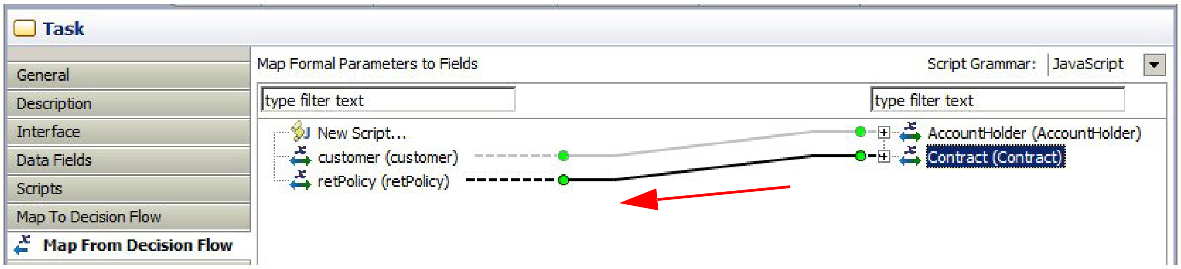
This specifies how data is mapped from the BPM process to the decisions flow, and from the decisions flow to the BPM process.

1. Select the decisions service task in the BPM process.
2. In the **Properties** view, select the **Map To Decision Flow** tab. This tab displays two columns of parameters — those on the left are the parameters defined in the BPM process; those on the right are the parameters defined in the decisions flow.
3. Map the parameters by placing the cursor on a parameter in the left column, and dragging it to the corresponding parameter in the right column.

4. Repeat the previous step for all of the parameters. For example:



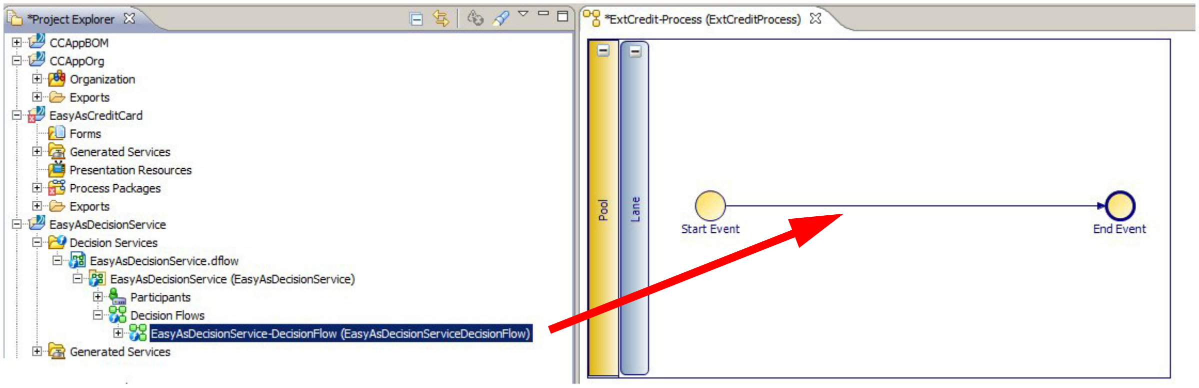
5. Select the **Map From Decision Flow** tab. This tab displays two columns of parameters — those on the left are the parameters defined in the decisions flow; those on the right are the parameters defined in the BPM process.
6. Map the parameters by placing the cursor on a parameter in the right column, and dragging it to the corresponding parameter in the left column.
7. Repeat the previous step for all of the parameters. For example:



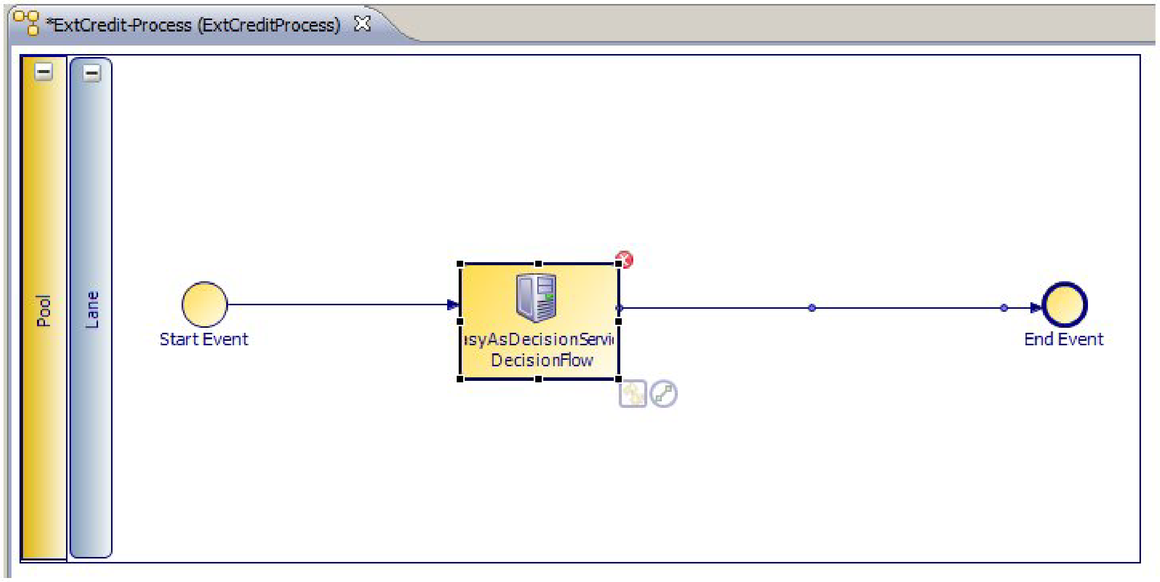
The configuration of the decisions service task is complete.

Drag-and-Drop a Decisions Flow onto a BPM Process

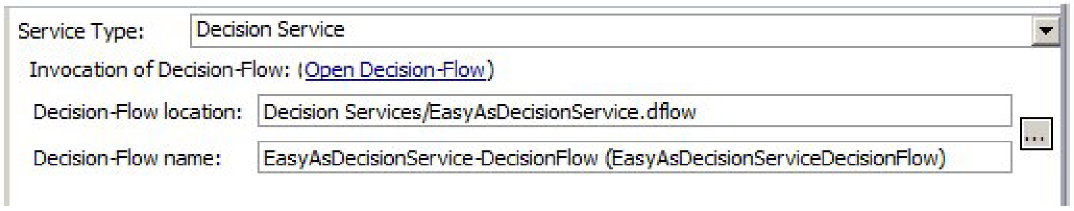
Another method of adding a decisions service task to a BPM process is to drag-and-drop a decisions flow onto the process:



This adds a decisions service task, for the selected decisions flow, to the BPM process:



The **General** tab in the **Properties** view for the decisions service task shows the decisions flow that will be invoked by the task:



Service Type: Decision Service

Invocation of Decision-Flow: (Open Decision-Flow)

Decision-Flow location: Decision Services/EasyAsDecisionService.dflow

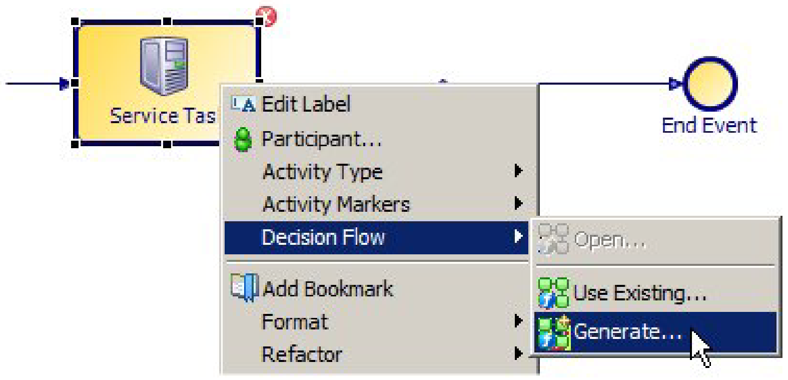
Decision-Flow name: EasyAsDecisionService-DecisionFlow (EasyAsDecisionServiceDecisionFlow)

After the decisions service task is added in this way, you will still need to define parameters in the BPM process (as described in Define Parameters in the BPM Process on page 80), then map the parameters between the BPM process and the decisions flow (as described in Map Parameters To and From the Decisions Service on page 80).

Generate a Decisions Flow from a BPM Process

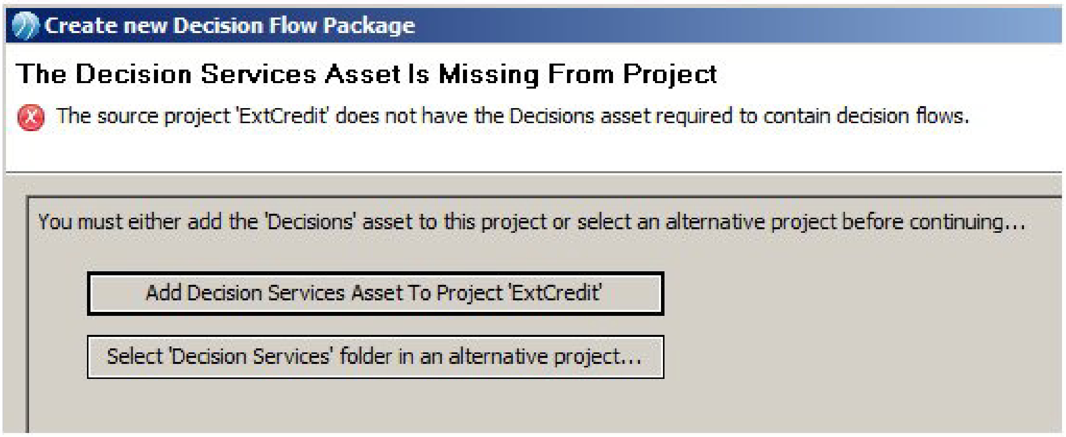
Decisions flows can also be generated from a BPM process. This is done in the following way:

1. From the Task Palette, drag a service task to your BPM process.
2. On the **General** tab for the service task, select **Decision Service** from the **Service Type** field drop-down list.
3. Right click on the newly added decisions service task and select **Decision Flow** > **Generate**:



If your current project has a **Decision Services** folder (that is, "Decision Services" was selected on the **Asset Type Selection** dialog when your project was originally created), the generated decisions flow is by default added to the **Decision Services** folder in the current project, and the **Decision Flow Package** dialog displays — proceed to step 4.

If "Decision Services" had not been selected on the **Asset Type Selection** dialog when your project was originally created, the following dialog is displayed:



Create new Decision Flow Package

The Decision Services Asset Is Missing From Project

The source project 'ExtCredit' does not have the Decisions asset required to contain decision flows.

You must either add the 'Decisions' asset to this project or select an alternative project before continuing...

Add Decision Services Asset To Project 'ExtCredit'
Select 'Decision Services' folder in an alternative project...

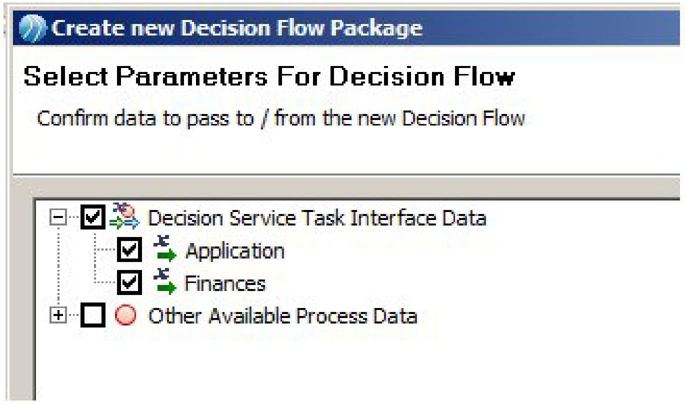
This dialog gives you the option of either adding a decisions services asset to the current project, or to point to a **Decision Services** folder in another project, if you want the generated decisions flow to be created in another project.

Choose where to generate the decisions flow by clicking on the appropriate button in the dialog above.

The **Decision Flow Package** dialog displays.

4. On the **Decision Flow Package** dialog, specify a name for the decisions flow by modifying the **File** field, then click **Next**.

   The **Select Parameters For Decision Flow** dialog displays:



Create new Decision Flow Package

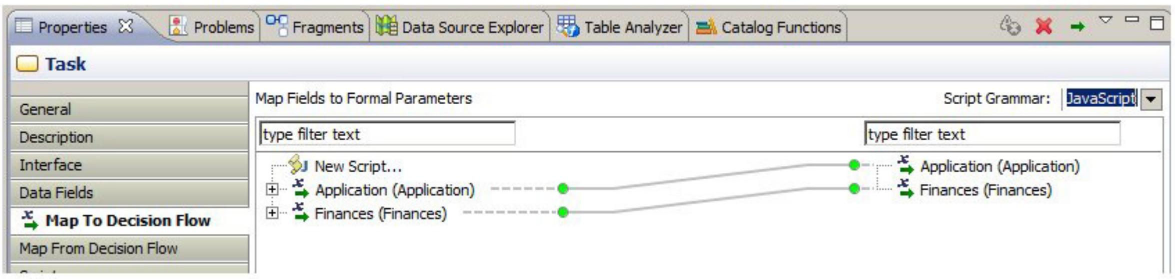
Select Parameters For Decision Flow

Confirm data to pass to / from the new Decision Flow

Decision Service Task Interface Data
Application
Finances
Other Available Process Data

5. Select parameters for the decisions flow, as follows:

   — **Decision Service Task Interface Data** - This option shows the parameters that are currently defined in the BPM process. You can choose some or all of these parameters for automatic inclusion in the created decisions flow. Choosing these parameters also causes them to be automatically mapped. For example, if you choose this option, after the decisions flow is generated, the **Map To Decision Flow** tab for the decisions service task appears as follows:



   — **Other Available Process Data** - Choosing this option means that parameters must be manually added to the generated decisions flow, then also manually mapped from and to the BPM process.

   For information about mapping parameters, see Map Parameters To and From the Decisions Service on page 80.

   After making a parameter selection, click **Next**. The **Package Information** dialog displays.

6. From the **Package Information** dialog, review the information, then click **Next**.

   The **Select Template** dialog displays.

7. Select the desired template, as follows:

   — **None type Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "None". This essentially makes the decisions service private (that is, it is not publicly exposed as a web service) — it can *only* be called from a decisions service task inside of a BPM process. When this template is chosen, a WSDL is *not* created in the **Generated Services** folder in Project Explorer.

   — **Start and End** - Choosing this template causes the decisions flow start event *trigger type*, and the end event *result type*, to be "Message". When this template is chosen, a WSDL is created in the **Generated Services** folder in Project Explorer, allowing the decisions service to be publicly exposed and consumed as a web service.

   After selecting the desired template, click **Finish**.

A new decisions flow is created, and it is opened in the decisions flow editor.

Chapter 7 Exposing a Decisions Service as a Web Service

This chapter provides information about exposing a decisions service as a web service.

Topics

The Default Generated Web Service Operation

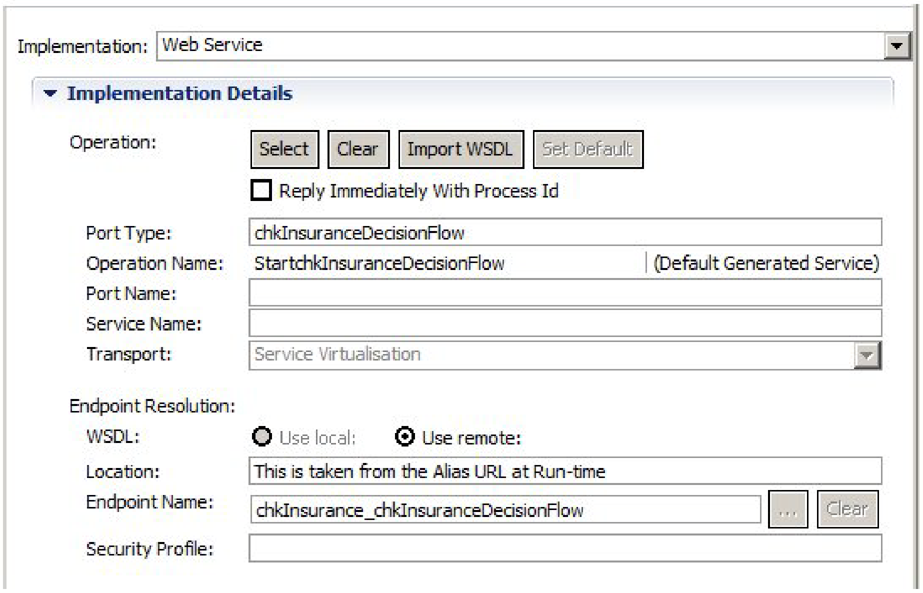
A decisions service is exposed as a web service when the decisions flow start event trigger type, and end event result type, are specified as "Message". For more information, see Decisions Flow Start and End Events on page 34.

When a decisions service is configured to be exposed as a web service, TIBCO Business Studio automatically creates a default web service operation, and does the following:

- It creates a WSDL file in the project's **Generated Services** folder. The WSDL filename is *DflowPackageName*.**wsdl**. The WSDL file contains:
  - — a portType with the same name as the decisions flow.
  - — an operation with the same name as the start event for the decisions flow.
  - — input and output messages and parameters that match the formal parameters and their modes defined in the decisions flow (see Adding Parameters to a Decisions Flow on page 38).

The **Generated Services** folder and WSDL file are not created until you save the project. A problem marker, stating that "The web service operation has not been generated", is displayed for the start and end events until you do this.

- It populates the **Operation** and **Endpoint Resolution** fields on the **General** tab in the **Properties** view for the start event of the decisions flow:



Implementation: Web Service

Implementation Details

Operation: Select | Clear | Import WSDL | Set Default

☐ Reply Immediately With Process Id

Port Type: chkInsuranceDecisionFlow

Operation Name: StartchkInsuranceDecisionFlow (Default Generated Service)

Port Name:

Service Name:

Transport: Service Virtualisation

Endpoint Resolution:

WSDL: ○ Use local: ● Use remote:

Location: This is taken from the Alias URL at Run-time

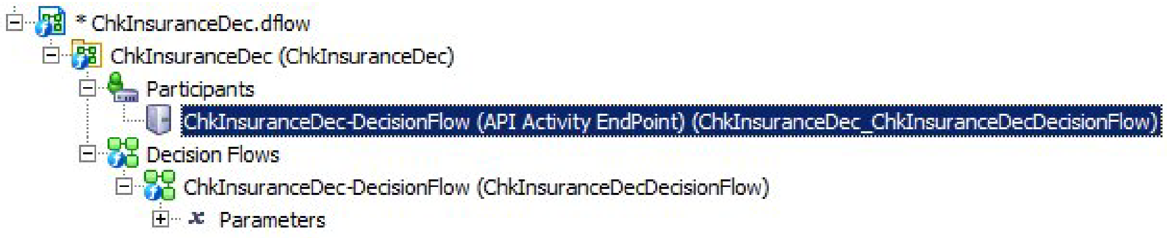
Endpoint Name: chkInsurance_chkInsuranceDecisionFlow ... Clear

Security Profile:

- Binding information that defines the web service endpoint is configured on the System Participant that was generated when the decisions service project was created. For information about the service endpoint, see Decisions Service Binding Configuration on page 92.

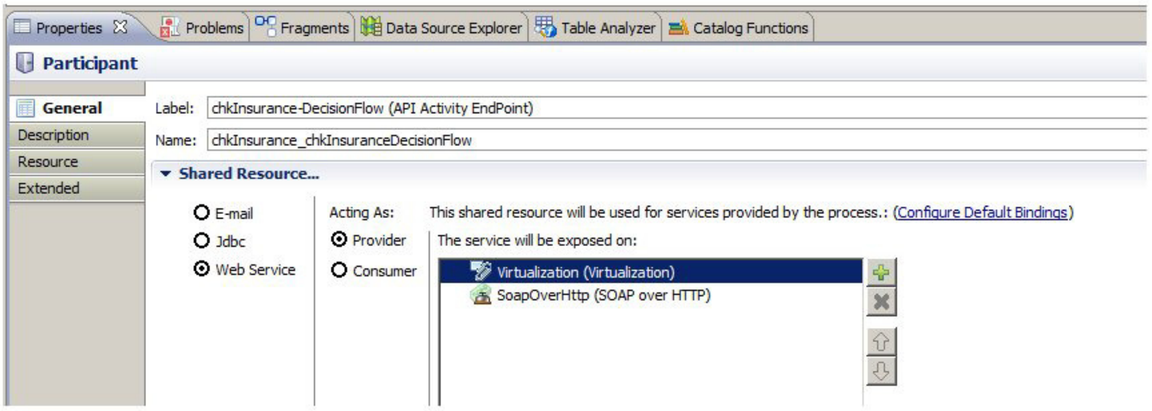
Decisions Service Binding Configuration

The decisions service endpoint is configured on the System Participant. The System Participant is created automatically when the decisions service project is created.



```
* ChkInsuranceDec.dflow
  ChkInsuranceDec (ChkInsuranceDec)
    Participants
      ChkInsuranceDec-DecisionFlow (API Activity EndPoint) (ChkInsuranceDec_ChkInsuranceDecDecisionFlow)
    Decision Flows
      ChkInsuranceDec-DecisionFlow (ChkInsuranceDecDecisionFlow)
        Parameters
```

Select the participant, then view the **General** tab on the **Properties** view. It will appear as follows:



Label: chkInsurance-DecisionFlow (API Activity EndPoint)

Name: chkInsurance_chkInsuranceDecisionFlow

Shared Resource...

( ) E-mail
( ) Jdbc
[✓ Web Service]

Acting As:
[✓ Provider]
( ) Consumer

This shared resource will be used for services provided by the process.: (Configure Default Bindings)

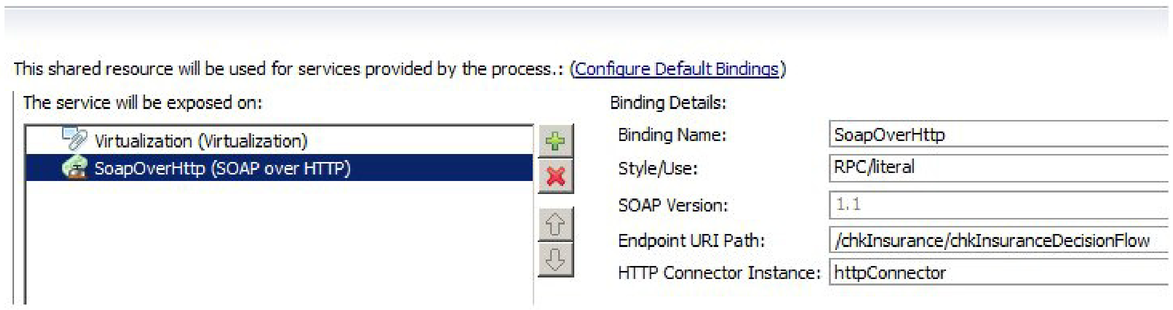
The service will be exposed on:

Virtualization (Virtualization)
SoapOverHttp (SOAP over HTTP)

Decision services support binding types of Virtualization, SOAP over HTTP, and SOAP over JMS.

Configuring SOAP Over HTTP Binding Type

Click the **SoapOverHttp** selection to view the binding details for SOAP over HTTP:



This shared resource will be used for services provided by the process.: (Configure Default Bindings)

The service will be exposed on:

- Virtualization (Virtualization)
- SoapOverHttp (SOAP over HTTP)

Binding Details:

Binding Name:	SoapOverHttp
Style/Use:	RPC/literal
SOAP Version:	1.1
Endpoint URI Path:	/chkInsurance/chkInsuranceDecisionFlow
HTTP Connector Instance:	httpConnector

The fields that can be modified in the **Binding Details** section for SOAP over HTTP are as follows:

- **Style/Use** - The default is to use RPC/literal. You can change this to Document/literal, however, Document/literal is only supported when a decisions flow has a single parameter; if it has more than one parameter, RPC/literal must be used.
- **EndPoint URI Path** - This can be changed to assign a new endpoint.
- **HTTP Connector Instance** - This is the logical name to identify the HTTP Connector resource instance in the decisions runtime that will be used to expose the service to client applications.

  The default value is **httpConnector**, which is the name of the default resource instance used by the decisions runtime to expose any services to clients over an HTTP connection.

  If you want to use a different HTTP Connector resource instance, you can do so using either of the following methods:

  — **Early binding** - Replace the name here with the name of a suitable HTTP Connector resource instance that already exists on the decisions runtime. (The mapping to the HTTP Connector resource instance will then be done automatically when you deploy the application to the decisions runtime.)

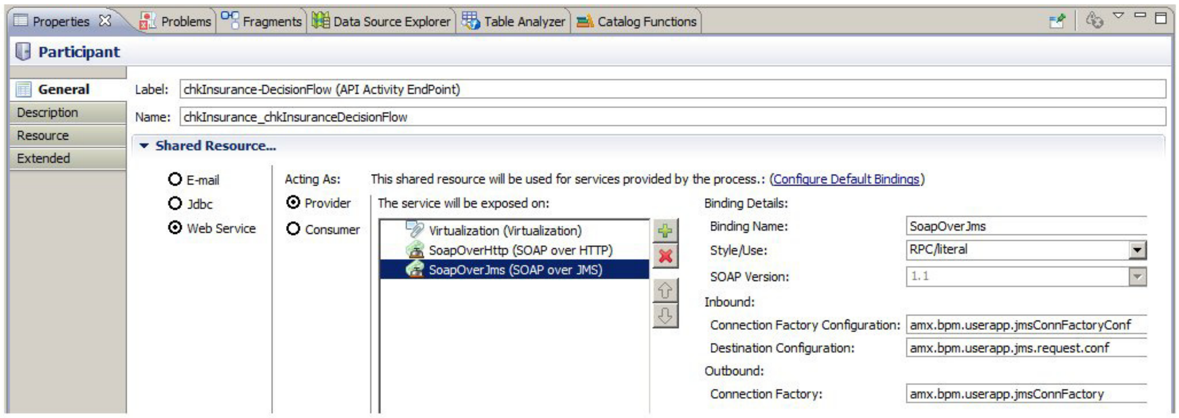
  — **Late binding** - Change or create the HTTP Connector resource instance to be used when you deploy the application to the decisions runtime. You do this by changing the value assigned to the **HttpInboundConnectionConfig** property on the **Property Configuration** page when deploying a decisions project or DAA. For details about deploying projects, see the TIBCO Business Studio documentation.

  If you export the decisions service to a DAA for subsequent deployment via TIBCO ActiveMatrix Administrator, an HTTP Connector resource instance will need to be configured using TIBCO ActiveMatrix Administrator. See the TIBCO ActiveMatrix Administrator documentation for more information about this.

Using a binding type of SOAP over JMS requires additional configuration. See Configuring a SOAP Over JMS Binding Type on page 94.

Configuring a SOAP Over JMS Binding Type

By default, **SoapOverJms** is not shown as one of the binding types. To add it, click the green plus icon to the right of the **This service will be exposed on** section and select **SoapOverJms** from the **Add Binding** dialog. Select **SoapOverJms** to view the binding details:



Label: chkInsurance-DecisionFlow (API Activity EndPoint)

Name: chkInsurance_chkInsuranceDecisionFlow

E-mail Jdbc [✓ Web Service]

Acting As: [✓ Provider] Consumer

This shared resource will be used for services provided by the process.: (Configure Default Bindings)

The service will be exposed on:
- Virtualization (Virtualization)
- SoapOverHttp (SOAP over HTTP)
- SoapOverJms (SOAP over JMS)

Binding Details:

Binding Name:	SoapOverJms
Style/Use:	RPC/literal
SOAP Version:	1.1
Inbound:	
Connection Factory Configuration:	amx.bpm.userapp.jmsConnFactoryConf
Destination Configuration:	amx.bpm.userapp.jms.request.conf
Outbound:	
Connection Factory:	amx.bpm.userapp.jmsConnFactory

Default resource template values are provided for the connection factory configuration, destination configuration, and connection factory. Change the values in these fields if custom resource templates have been created.

Note that on an ActiveMatrix BPM system, resource templates exist by default for the values shown here.

Also note that if multiple parameters are used in the decisions service, ensure that "RPC/literal" is selected in the **Style/Use** field.

Additional configuration for SOAP over JMS is required at deployment time. For information, see Deployment When Using SOAP Over JMS on page 110.

Separating the BOM from the Decisions Service Project

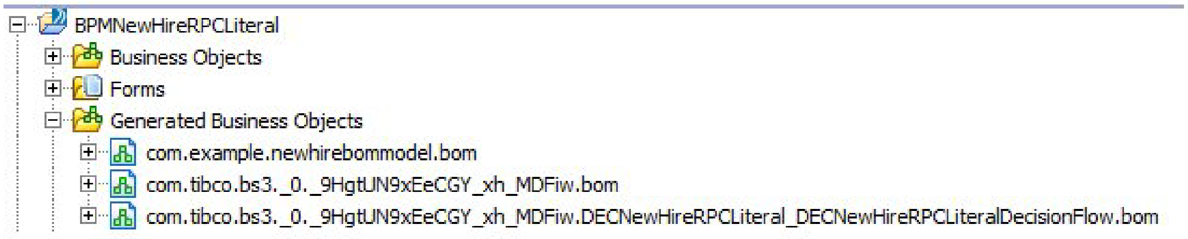
The information in this section is applicable only if you are:

- developing a decisions service that will be called from an ActiveMatrix BPM process,
- calling the decisions service as a "web service", and
- using a binding type of SOAP over JMS.

When calling a decisions service as a web service, there is an assumption that the service has been developed externally. Therefore, the concrete WSDL defines the contract and must be imported. Also, a BPM project is generated and populated with the data types in the WSDL. However, if the decisions service project and BOM are available, you will need to remove these from the workspace (or close these projects) *before* configuring the BPM process to call the web service.

If the BOM that is used by the BPM process is in the same workspace as the decisions service project, there would be a conflict between that BOM and the BOM objects generated when the WSDL is imported.

The following shows the BOM objects that are generated when you import the WSDL into your decisions service project:


BPMNewHireRPCLiteral
Business Objects
Forms
Generated Business Objects
com.example.newhirebommodel.bom
com.tibco.bs3._0._9HgtUN9xEeCGY_xh_MDFiw.bom
com.tibco.bs3._0._9HgtUN9xEeCGY_xh_MDFiw.DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow.bom

Restricting Runtime Access to Decisions Services

Authentication requirements that will restrict access to a decisions service depends on the client calling the decisions service as follows:

- **External client application** - Currently, there are no security policies applied to the decisions service endpoint when it is exposed as a web service. This means that the only way to restrict access is by running in a secure environment (that is, behind firewalls, etc.).
- **BPM application** - In a decisions flow, the start event should have a trigger type of "None" (not "Message") so that the decisions service will not be exposed as a web service. The login credentials used to access the calling BPM process will ensure that unauthorized users cannot access the decisions service.

Chapter 8 Decisions Service Deployment

This chapter describes deployment of decisions services.

Topics

Overview

Three types of applications that you will typically want to deploy are:

- **Decisions service application** - This application contains the decisions flow; it is a service provider.
- **BPM decisions extension application** - This is a BPM application that contains a BPM service task with a type of *decisions service*; it is a service consumer.
- **BPM process** - A BPM process that consumes a web service. The BPM service task is configured with a type of *web service.* The WSDL from the decisions service is used as the contract for the service call.

Deployment Scenarios

After completing development of these types of applications in TIBCO Business Studio, they must be deployed to a runtime environment. This can be done in a number of ways:

- Directly from TIBCO Business Studio using a deployment server - see Deploying from TIBCO Business Studio on page 99.
- Export a Distributed Application Archive (DAA), then deploy using TIBCO ActiveMatrix Administrator - see Deploying from TIBCO ActiveMatrix Administrator on page 101.
- As part of a "Deploy Project", from which you generate deployment scripts. These scripts allow for a *scripted deployment* via a command line - see Deploying Using a Deployment Script on page 105.
- Using the Generate Administrator CLI Script wizard. This is done by right-clicking on a DAA in the TIBCO Business Studio Project Explorer, then selecting **Generate Administrator CLI Script**. You can either generate the script, *or* generate the script *and* execute it after it is generated. For more information about using this wizard, see the *TIBCO ActiveMatrix BPM SOA Composite Development* guide.

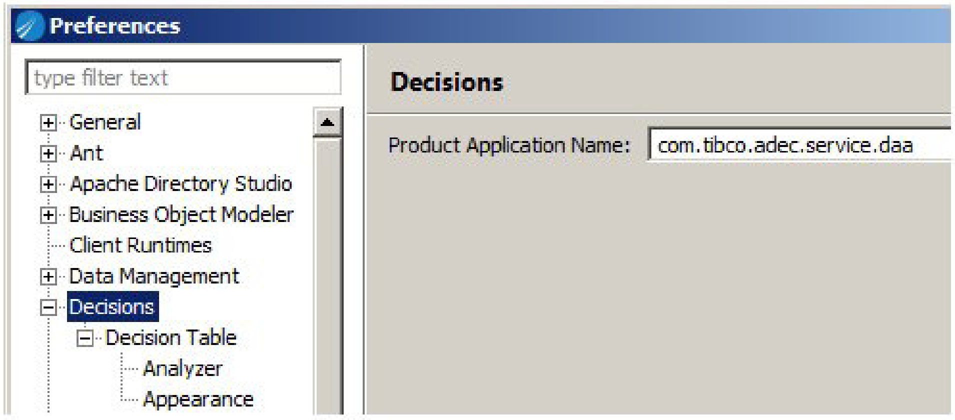
Note that if you are exposing a decisions service as a web service, and using SOAP over JMS as a binding type, additional configuration is required during deployment of your decisions service application and BPM process. For more information, see Deployment When Using SOAP Over JMS on page 110.

Deploying from TIBCO Business Studio

You can deploy directly from TIBCO Business Studio using a deployment server. This type of deployment is more often done in a development/testing environment. In a production environment, it is more typical to export a DAA, then deploy from TIBCO ActiveMatrix Administrator using the DAA.

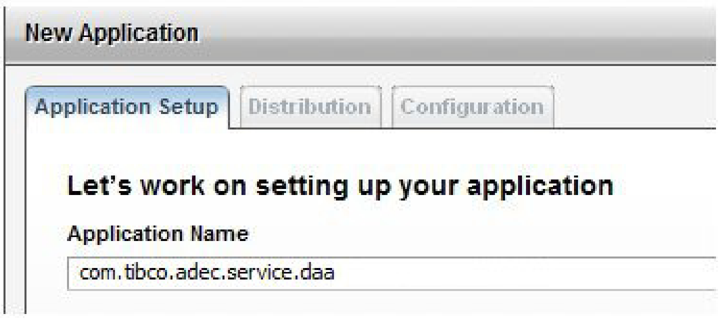
To deploy a project from TIBCO Business Studio, you must first ensure that the **Product Application Name** specified on the **Preferences** > **Decisions** dialog in TIBCO Business Studio (see illustration below) is the same **Product Application Name** that was specified when the ActiveMatrix Decisions IT was configured.

To specify the **Product Application Name** in TIBCO Business Studio, select **Window** > **Preferences** > **Decisions** and enter the name in the **Product Application Name** field. The name defaults to "com.tibco.adec.service.daa":



Product Application Name: com.tibco.adec.service.daa

The **Product Application Name** specified on the **Preferences** > **Decisions** dialog *must* match the **Product Application Name** specified in TIBCO ActiveMatrix Administrator when the ActiveMatrix Decisions Service IT Application is configured.



Let's work on setting up your application

Application Name: com.tibco.adec.service.daa

Configuration of the ActiveMatrix Decisions Service IT Application is done after TIBCO ActiveMatrix Decisions is installed on a runtime node, which is described in the *TIBCO ActiveMatrix Decisions Installation* guide.

After ensuring the correct name is specified on the **Preferences** > **Decisions** dialog, you can deploy the project using a deployment server. For information about creating and using a deployment server, see the TIBCO Business Studio documentation.

Deploying from TIBCO ActiveMatrix Administrator

A decisions service project can be deployed from ActiveMatrix Administrator using a Distributed Application Archive (DAA). For information about deployment from ActiveMatrix Administrator, see the *TIBCO ActiveMatrix Administration* guide.

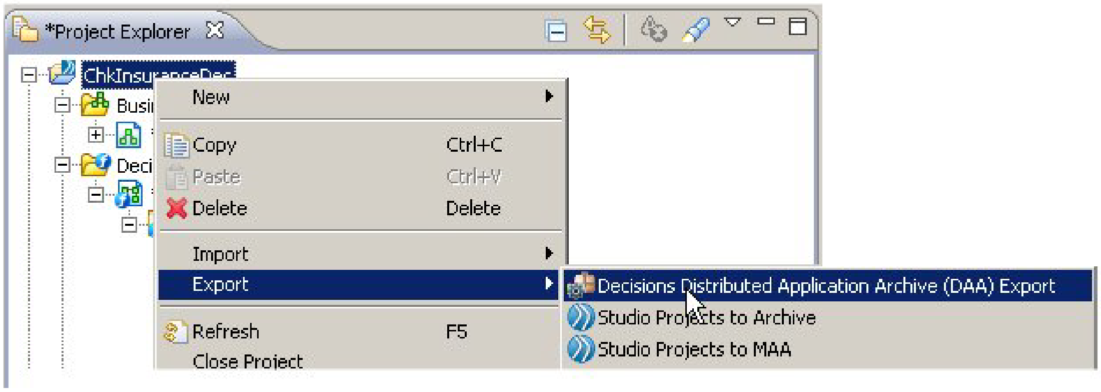
To deploy a decisions project from ActiveMatrix Administrator, you must first generate a DAA for your decisions project. There are a couple of ways you can do that:

- Export the DAA directly from TIBCO Business Studio — see Exporting a Decisions Project DAA from TIBCO Business Studio on page 101
- Generate the DAA from the command line using an Apache Ant Task — see Generating a DAA From the Command Line on page 102

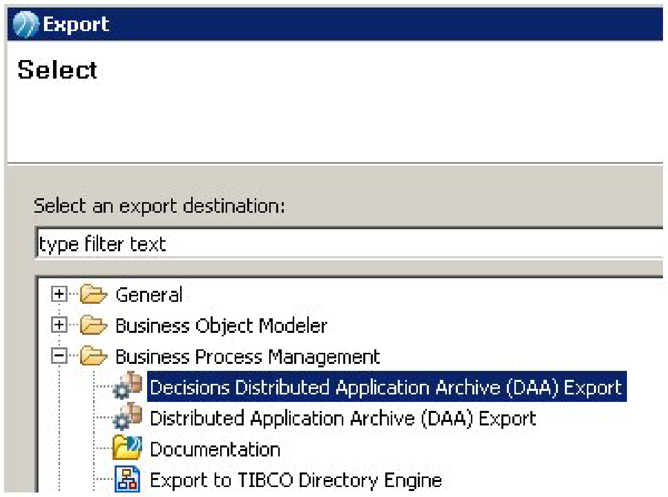
Exporting a Decisions Project DAA from TIBCO Business Studio

You can export a DAA for a decisions service project from TIBCO Business Studio in the following ways:

— Right-click on the project in Project Explorer, then select **Export** > **Decisions Distributed Application Archive (DAA)**:



— Or select **File** > **Export**, then choose **Business Process Management** > **Decisions Distributed Application Archive (DAA)** on the **Export** dialog:


Export

Select

Select an export destination:

type filter text

General
Business Object Modeler
Business Process Management
Decisions Distributed Application Archive (DAA) Export
Distributed Application Archive (DAA) Export
Documentation
Export to TIBCO Directory Engine

Prior to deployment or export to a DAA, you should check the **Problems** tab to ensure there are no errors in the project. You shouldn't rely on only problem markers in Project Explorer, as some problem markers may not be visible if the entire structure is not expanded.

Generating a DAA From the Command Line

Project DAAs can be built from the command line using Ant scripts.

For general information about generating DAAs from the command line, see:

- "Generating a DAA from the Command Line using an Ant Task" in the *TIBCO Business Studio BPM Implementation Guide*.
- "TIBCO Business Studio Command-Line Interface" in the *TIBC0 ActiveMatrix BPM SOA Composite Development* guide.

When the TIBCO Business Studio ActiveMatrix Decisions Add-in is installed, an "sda.generateDAA" Ant task is available to generate DAAs for decisions projects.

The following provides a summary of the things you need to do to generate DAAs that include decisions projects:

1. Create a build file (for example, build.xml) that includes the sda.generateDAA Ant task.

   The build file should be similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="DecisionsProject" default="default">
    <description>
            DAA Generation of Decisions Projects
    </description>
    <target name="default" description="DAA Generation of Decisions Projects">
      <tbs.importProjects dir="ProjectSource"/>
      <bpm.generateDAA buildbeforegenerating="false" daalocation="DAA_Dir"/>
      <sda.generateDAA buildbeforegenerating="false" daalocation="DAA_Dir"/>
    </target>
</project>
```

where:

— *ProjectSource* - The location of the TIBCO Business Studio projects for which DAAs are to be generated.

— *DAA_Dir* - The directory in which the DAAs are to be generated.

The **buildbeforegeneration** attribute should be set to "false" at this time. Setting it to "true" can cause problems under some circumstances in this release.

2. Save the build file in a directory, which you will reference when you execute the DAA generation.
3. Ensure your TIBCO Business Studio projects are in *ProjectSource* as referenced in the build file.
4. Create a directory in which the generated DAAs will be placed. This is also referenced in the build file (*DAA_Dir*).
5. Create an *empty* directory that serves as a workspace for the DAA generation. This directory is referenced when you execute the DAA generation.

6. Generate the DAAs by executing `amx_eclipse_ant.exe`, as follows:
   a. Open a command prompt.
   b. Change directory to:

      Studio_Home\studio\3.6\eclipse

      where *Studio_Home* is the directory in which TIBCO Business Studio is installed.
   c. Execute the build using the following general format:

      ```
      amx_eclipse_ant.exe -f <build file> -data <workspace directory>
      ```

      For example:

      ```
      amx_eclipse_ant.exe -f c:\tmp\build.xml -data c:\tmp\daa-ws
      ```

When the build completes, the DAAs will be located in the directory specified in the **daalocation** attribute in the build file.

.A build log file is generated in `<workspace dir>\.metadata\.log`.

You can also redirect console output to a file. For example:

```
amx_eclipse_ant.exe -f c:\tmp\build.xml -data c:\tmp\daa-ws > c:\tmp\output.txt
```

Deploying Using a Deployment Script

TIBCO Business Studio provides the ability to generate a deployment script that can be used to deploy ActiveMatrix Decisions projects via a script from either TIBCO Business Studio or from the command line.

You must first create a "Deploy Project" in TIBCO Business Studio. From within the Deploy Project, you generate DAAs for the "resources" (ActiveMatrix BPM and/or ActiveMatrix Decisions projects) in the Deploy Project, then generate a build.xml file (an Apache Ant script). When the script is executed, the projects in the Deploy Project are deployed to the specified node.

For detailed information about creating and configuring a Deploy Project, see the *TIBCO ActiveMatrix BPM Deployment* guide.

Deployment Scripts for Decisions Projects

This section only provides information that is specific to generating and executing deployment scripts for deployment projects that contain decisions projects. For general information about creating a Deploy Project, see the *TIBCO ActiveMatrix BPM Deployment* guide.

You may need to refresh the deployment server so that Administrator is aware of the current state of the server.

And depending on your configuration, for your BPM project to successfully deploy, you may need to deploy your ActiveMatrix Decisions project first, refresh the server, then configure, generate scripts, and deploy your BPM project. (Note that this may not be needed in all situations. It may, however, be required if you are using virtualized bindings.)

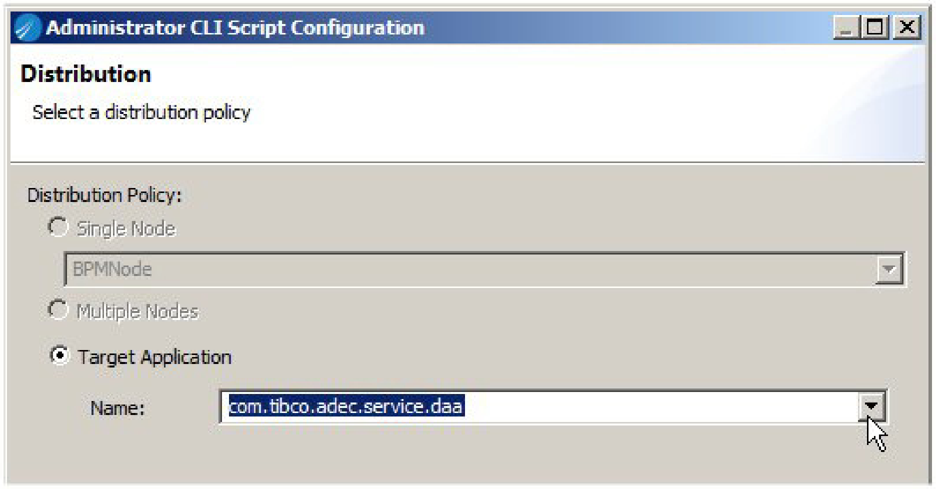
For more information, see the *TIBCO ActiveMatrix BPM Deployment* guide.

Multiple Target Applications

If your Deploy Project contains both ActiveMatrix BPM and ActiveMatrix Decisions projects, you must specify a *target application* that is appropriate for each of the resources / applications in the Deploy Project.

- The target application for ActiveMatrix BPM projects defaults to **amx.bpm.app**.
- The target application for ActiveMatrix Decisions projects defaults to **com.tibco.adec.service.daa**.

The target application is specified for each resource in the Deploy Project during the configuration phase on the Distribution dialog:



Distribution

Select a distribution policy

Distribution Policy:

( ) Single Node

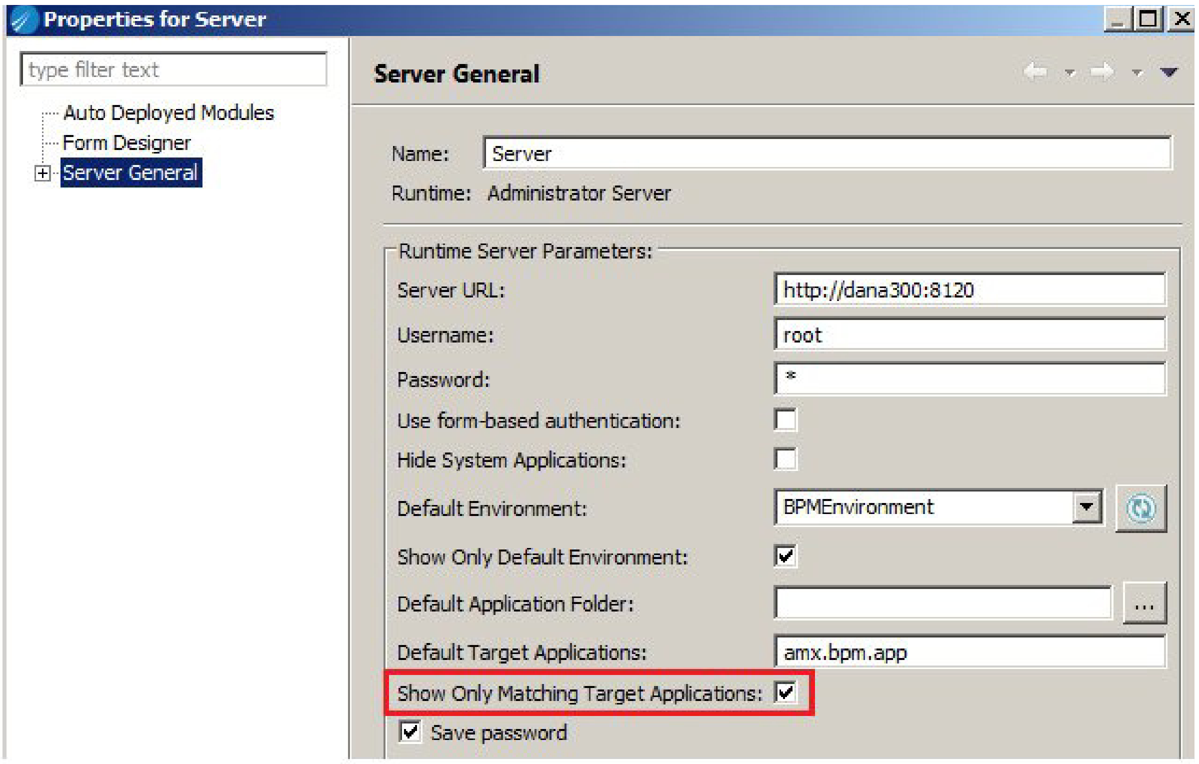
BPMNode

( ) Multiple Nodes

(●) Target Application

Name: com.tibco.adec.service.daa

The **Target Application Name** field will default to the target application whose capability matches those of the resource being configured if the **Show Only Matching Target Applications** check box is selected on the deployment server's Server General dialog — see below (it is selected by default).



Name: Server
Runtime: Administrator Server

Runtime Server Parameters:

Server URL:	http://dana300:8120
Username:	root
Password:	*
Use form-based authentication:	☐
Hide System Applications:	☐
Default Environment:	BPMEnvironment
Show Only Default Environment:	✓
Default Application Folder:	
Default Target Applications:	amx.bpm.app
Show Only Matching Target Applications:	✓

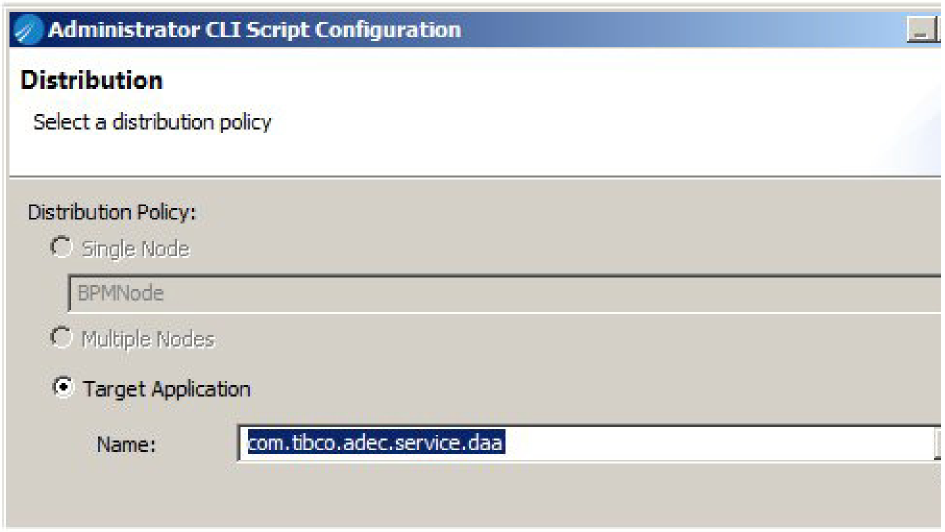
✓ Save password

To be able to set multiple target applications in a Deploy Project, you must have ActiveMatrix Platform 3.3.0 Hotfix 5 installed. This hotfix provides the ability to select the target application on the deployment project wizard Distribution dialog, plus it adds the **Show Only Matching Target Applications** check box to the deployment server properties.

This hotfix must be installed in both the development environment (TIBCO Business Studio) and the runtime (Administrator Server).

Also note that when installing the hotfix, ensure that you install *all* of the available features. The **ActiveMatrix Platform 3.3.0 HF5** feature *must* be installed (even if you do not have ActiveMatrix Platform installed) on your system.

On the **Administrator CLI Script Configuration** dialog during Deploy Project configuration, ensure that the target application for Decisions projects is set to **com.tibco.adec.service.daa**:


Administrator CLI Script Configuration

Distribution

Select a distribution policy

Distribution Policy:

Single Node [ ]

BPMNode

Multiple Nodes [ ]

[✓ Target Application]

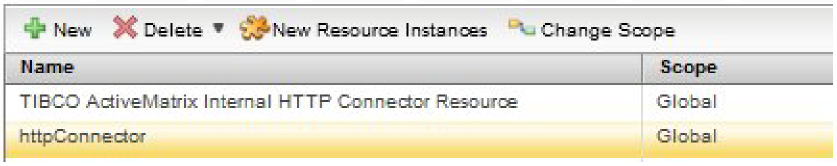
Name: com.tibco.adec.service.daa

And for ActiveMatrix BPM projects, ensure that the target application is set to **amx.bpm.app**.

Shared HTTP Connector Resource

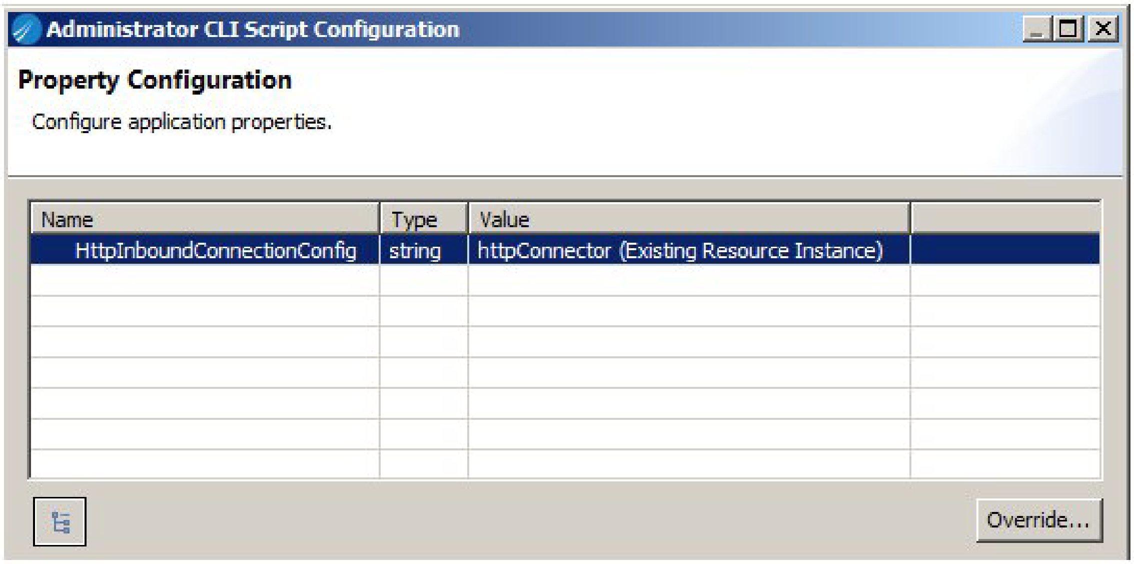
In almost all applications, the HTTP Connector that exposes the service endpoint is called "httpConnector" because this is the default value in TIBCO Business Studio. When this resource first gets installed to the node it is tied to a specific application. Then when another application using the same shared resource name is installed, it displays an error because of the scope of the connector (that is, it is not "Global").

Connect to the configured deployment server using ActiveMatrix Administrator and ensure that there is an httpConnector with a scope of Global:



Name	Scope
TIBCO ActiveMatrix Internal HTTP Connector Resource	Global
httpConnector	Global

And when configuring the resources in the Deploy Project Configuration Application wizard, ensure that the "httpConnector" is configured either for a "New Resource from Existing Template" or for an "Existing Resource Instance":


Administrator CLI Script Configuration
Property Configuration
Configure application properties.
Override...

Name	Type	Value
HttpInboundConnectionConfig	string	httpConnector (Existing Resource Instance)

If the httpConnector defaults to New Resource for this application, click the **Override** button to configure it to a New Resource from Existing Template.

Deployment When Using SOAP Over JMS

This section describes the process of deploying a decisions service that is exposed as a web service, and that uses SOAP over JMS.

1. Generate a DAA for the decisions service by right-clicking on the project in the TIBCO Business Studio Project Explorer, then selecting **Export** > **Decisions Distributed Application Archive (DAA)**.

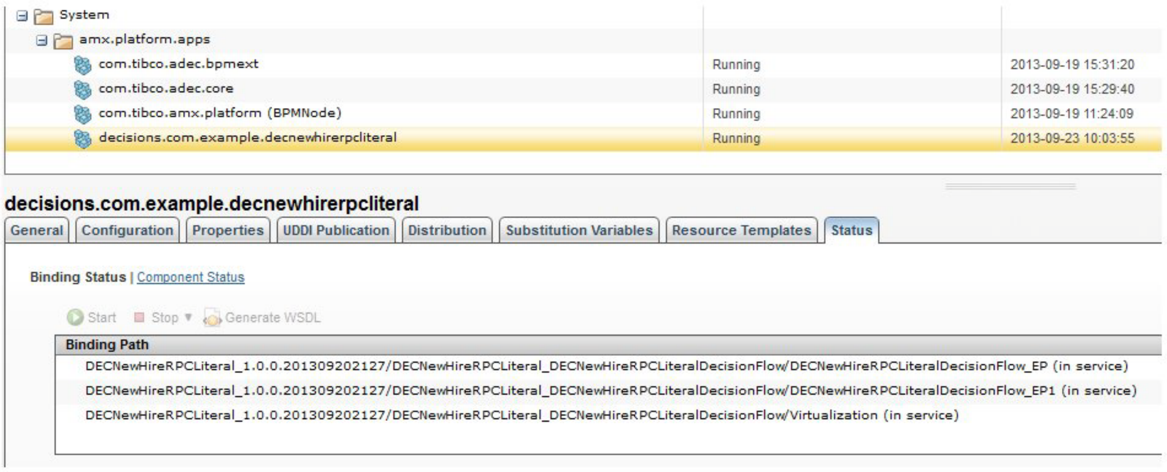
2. In TIBCO ActiveMatrix Administrator, add a new application for the decisions service and deploy it.

   During the deployment of the decisions service application, a **Supply values for configuration properties...** dialog is displayed, which is used to specify resource templates. You must select the same resource templates that were configured for the decisions service participant in TIBCO Business Studio. For more information about those resource templates, see Configuring a SOAP Over JMS Binding Type on page 94.

3. Generate a WSDL for the decisions service, as follows:

   a. Select the decisions service application that you have just deployed, then select the **Status** tab.

      This shows you the binding types:



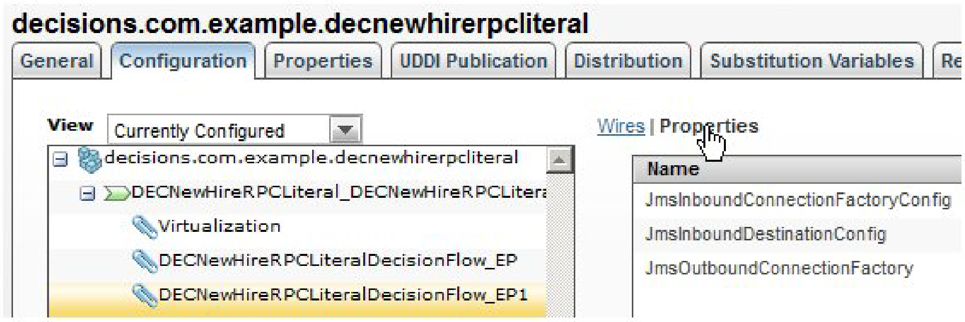
System		
amx.platform.apps		
com.tibco.adec.bpmext	Running	2013-09-19 15:31:20
com.tibco.adec.core	Running	2013-09-19 15:29:40
com.tibco.amx.platform (BPMNode)	Running	2013-09-19 11:24:09
decisions.com.example.decnewhirerpcliteral	Running	2013-09-23 10:03:55

decisions.com.example.decnewhirerpcliteral

Binding Path
DECNewHireRPCLiteral_1.0.0.201309202127/DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow/DECNewHireRPCLiteralDecisionFlow_EP (in service)
DECNewHireRPCLiteral_1.0.0.201309202127/DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow/DECNewHireRPCLiteralDecisionFlow_EP1 (in service)
DECNewHireRPCLiteral_1.0.0.201309202127/DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow/Virtualization (in service)

You need to select the binding type to generate the WSDL, but notice that it does not tell you which one is for SOAP over HTTP, and which one is for SOAP over JMS (it does show you which one is for virtualization).

You can determine which is which by selecting the **Configuration** tab, then drilling down in the decisions flow and clicking the **Properties** link in the right pane. For example:


decisions.com.example.decnewhirerpcliteral

- decisions.com.example.decnewhirerpcliteral
  - DECNewHireRPCLiteral_DECNewHireRPCLiter
    - Virtualization
    - DECNewHireRPCLiteralDecisionFlow_EP
    - DECNewHireRPCLiteralDecisionFlow_EP1

Name
JmsInboundConnectionFactoryConfig
JmsInboundDestinationConfig
JmsOutboundConnectionFactory

In this example the "EP1" entry is for SOAP over JMS.

b. On the **Status** tab, select the appropriate entry for SOAP over JMS in the **Binding Path** section, then click **Generate WSDL**.

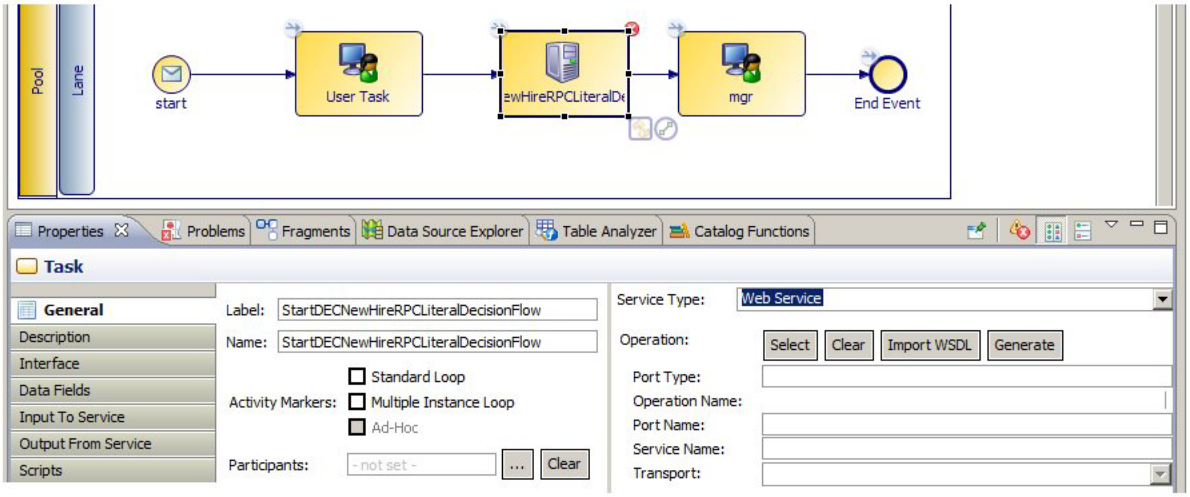
c. When the WSDL opens in the browser, copy the URL from the address line.

4. In TIBCO Business Studio, open the BPM service project.

   Note that when using SOAP over JMS, the BPM service project needs to be separate from the BOM project; that is, they cannot be in the same workspace.

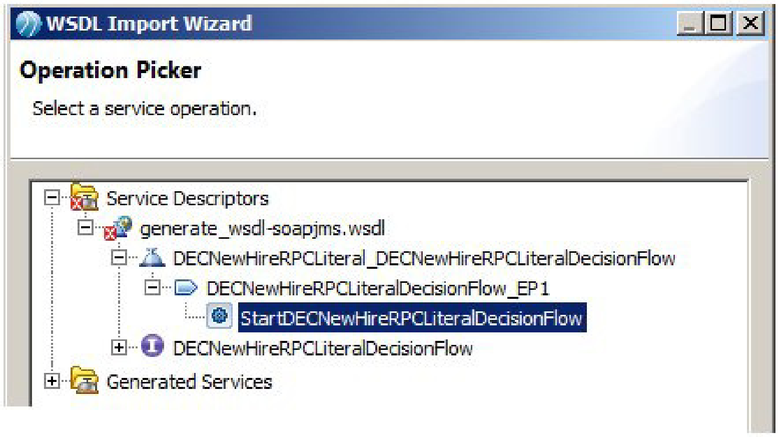
5. Select the decisions service task in the BPM process.

6. On the **General** tab, select "Web Service" from the **Service Type** field list:



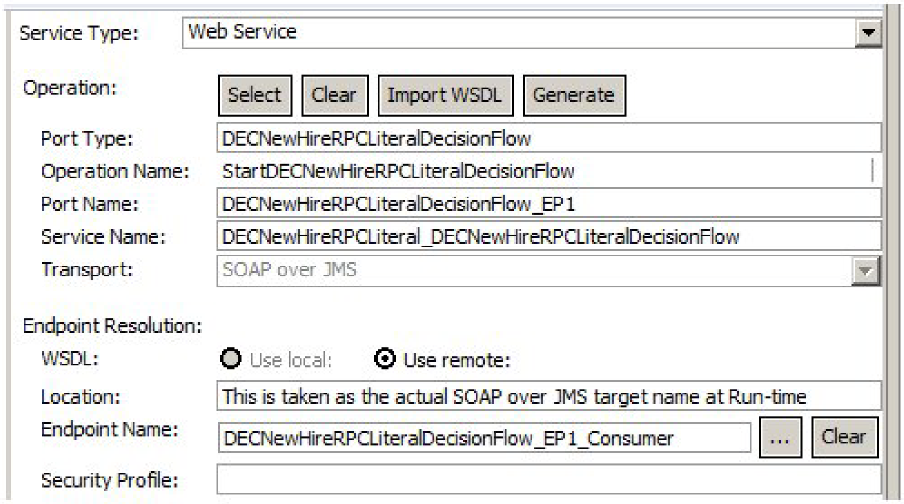
Label: StartDECNewHireRPCLiteralDecisionFlow
Name: StartDECNewHireRPCLiteralDecisionFlow
Activity Markers:
Participants: - not set -
Service Type: Web Service
Operation:
Port Type:
Operation Name:
Port Name:
Service Name:
Transport:

7. Click the **Import WSDL** button.
8. On the **WSDL Selection** dialog, choose **Import from a URL**, click **Next**, then paste, into the **URL** field, the URL you previously copied from the generated WSDL.
9. Click **Next** again, then on the **Operation Picker** dialog, select the operation with the icon and click **Finish**:



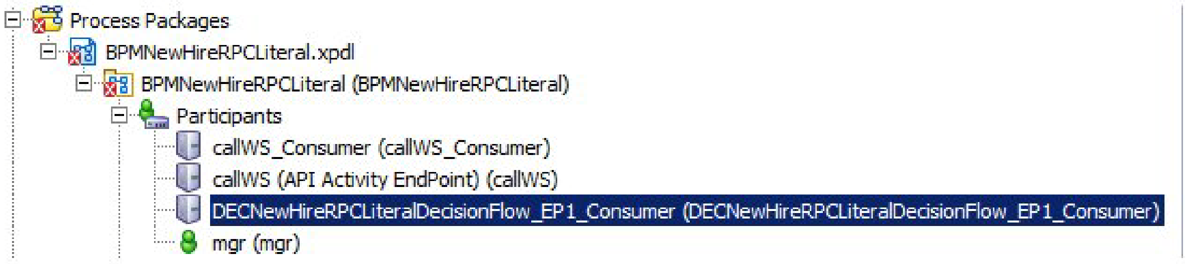
Service Descriptors
generate_wsdl-soapjms.wsdl
DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow
DECNewHireRPCLiteralDecisionFlow_EP1
StartDECNewHireRPCLiteralDecisionFlow
DECNewHireRPCLiteralDecisionFlow
Generated Services

The configuration for the web service is filled after the WSDL is imported, including the "Consumer" endpoint:



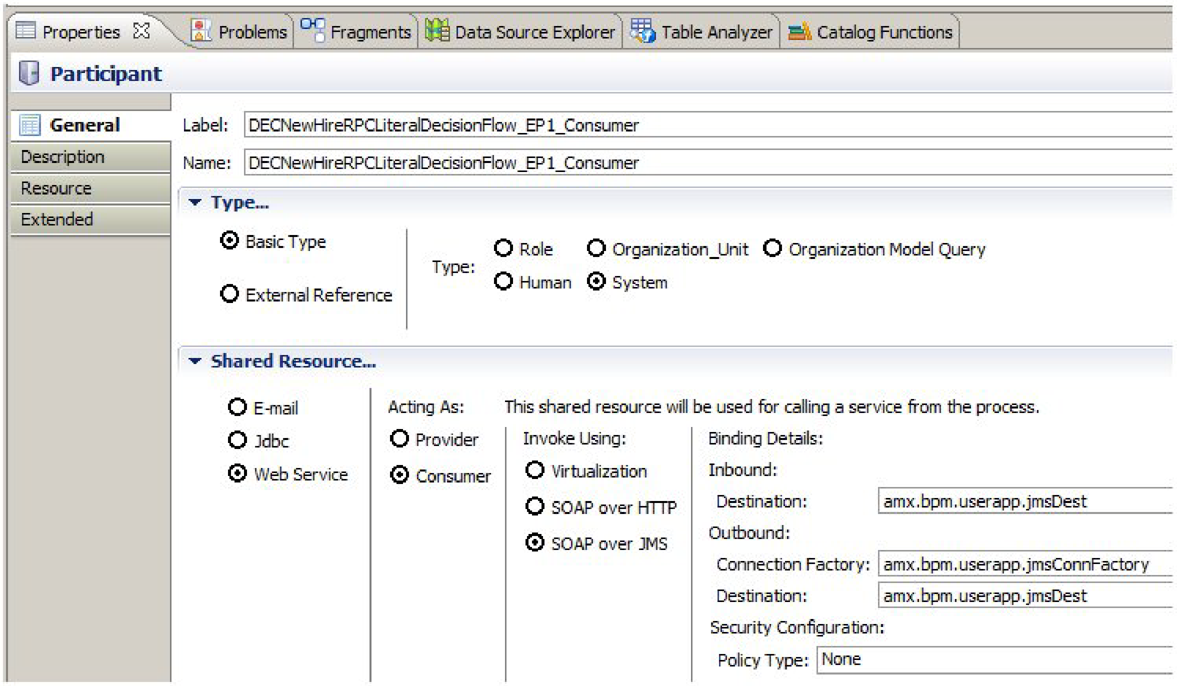
Service Type:	Web Service
Operation:	Select · Clear · Import WSDL · Generate
Port Type:	DECNewHireRPCLiteralDecisionFlow
Operation Name:	StartDECNewHireRPCLiteralDecisionFlow
Port Name:	DECNewHireRPCLiteralDecisionFlow_EP1
Service Name:	DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow
Transport:	SOAP over JMS
Endpoint Resolution:	
WSDL:	Use local: [✓ Use remote:]
Location:	This is taken as the actual SOAP over JMS target name at Run-time
Endpoint Name:	DECNewHireRPCLiteralDecisionFlow_EP1_Consumer ... Clear
Security Profile:	

A participant of the same name that is used for SOAP over JMS is also generated:



```
Process Packages
  BPMNewHireRPCLiteral.xpdl
    BPMNewHireRPCLiteral (BPMNewHireRPCLiteral)
      Participants
        callWS_Consumer (callWS_Consumer)
        callWS (API Activity EndPoint) (callWS)
        DECNewHireRPCLiteralDecisionFlow_EP1_Consumer (DECNewHireRPCLiteralDecisionFlow_EP1_Consumer)
        mgr (mgr)
```

If you select the generated participant, you can see the configuration on the **Properties** view **General** tab:



Label: DECNewHireRPCLiteralDecisionFlow_EP1_Consumer

Name: DECNewHireRPCLiteralDecisionFlow_EP1_Consumer

Type...

[✓ Basic Type] External Reference

Type: Role Organization_Unit Organization Model Query Human [✓ System]

Shared Resource...

E-mail Jdbc [✓ Web Service]

Acting As: Provider [✓ Consumer]

This shared resource will be used for calling a service from the process.

Invoke Using: Virtualization SOAP over HTTP [✓ SOAP over JMS]

Binding Details:

Inbound:

Destination: amx.bpm.userapp.jmsDest

Outbound:

Connection Factory: amx.bpm.userapp.jmsConnFactory

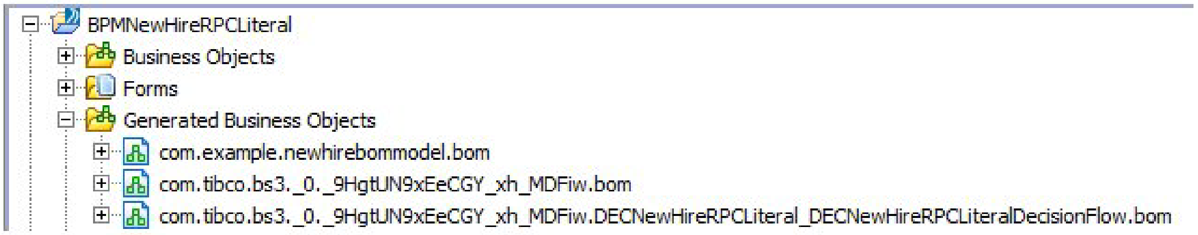
Destination: amx.bpm.userapp.jmsDest

Security Configuration:

Policy Type: None

Notice that instead of a destination configuration, a JMS destination resource template is specified in the **Inbound Destination** field in the **Binding Details** section. This application resource template does not exist by default, however. It must be created.

Another result of the WSDL import is the generation of the following BOM objects:



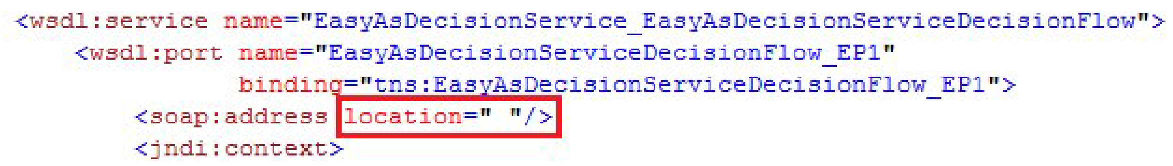
- BPMNewHireRPCLiteral
  - Business Objects
  - Forms
  - Generated Business Objects
    - com.example.newhirebommodel.bom
    - com.tibco.bs3._0._9HgtUN9xEeCGY_xh_MDFiw.bom
    - com.tibco.bs3._0._9HgtUN9xEeCGY_xh_MDFiw.DECNewHireRPCLiteral_DECNewHireRPCLiteralDecisionFlow.bom

Note that these BOM objects would have a conflict with the BOM objects in the decisions service project if you have the BOM project in the same workspace as the BPM service project. That is the reason the BOM project must be in a separate workspace from the BPM service project. When they are separate, there is no conflict.

10. Update the **location** attribute in the generated WSDL to resolve an error that is shown on the **Problems** tab, which indicates that an invalid location is specified in the WSDL:

    a. In the **Service Descriptors** folder, right-click the generated WSDL and select **Open With > WSDL Editor**.

    b. In the **soap:address location** attribute, enter any value (such as a space) inside the double quotes (they are empty by default):



```
<wsdl:service name="EasyAsDecisionService_EasyAsDecisionServiceDecisionFlow">
    <wsdl:port name="EasyAsDecisionServiceDecisionFlow_EP1"
               binding="tns:EasyAsDecisionServiceDecisionFlow_EP1">
        <soap:address location=" "/>
        <jndi:context>
```

    c. Save and close the WSDL file.

11. If the BPM process depends on an organization model, generate a DAA for the organization model project:

    — Right-click on the project in the TIBCO Business Studio Project Explorer, then select **Export** > **Decisions Distributed Application Archive (DAA)**.

    a. Then in TIBCO ActiveMatrix Administrator, add a new application for the organization model project and deploy it.

12. Generate a DAA for the BPM service project:

    — Right-click on the project in the TIBCO Business Studio Project Explorer, then select **Export** > **Decisions Distributed Application Archive (DAA)**.

13. In TIBCO ActiveMatrix Administrator, add a new application for the BPM service project.

    On the **Supply values for configuration properties...** dialog, if you clicked the eye dropper icon to the right of the inbound and outbound JMS destination properties, you would see that they are not configured because those application templates do not exist yet.

a. On the **Supply values for configuration properties...** dialog, click **Next** to proceed with the application creation.

   The next dialog should tell you that the validation was successful. Note, however, that the message will also state that there were a few issues but you will have an opportunity to resolve them after you click **Deploy**.

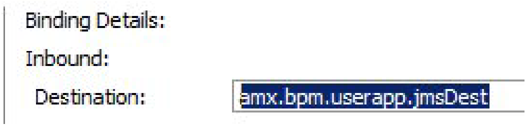
b. Click **Deploy**.

c. Click the **Resolve** link in the upper right part of the dialog.

d. Click the **new resource template** link in the upper right part of the dialog.

e. On the **Add Resource Template** dialog, choose "JMS Destination" from the **Type** field list.

f. Change the name in the **Name** field to the value that is in the **Inbound Destination** field for the SOAP over JMS participant configuration in Business Studio (see step 9):



Binding Details:
Inbound:
Destination: amx.bpm.userapp.jmsDest

g. Ensure the scope is set to "Global".

h. Click the eye dropper icon to the right of the **JNDI Connection Configuration** field, then choose the "amx.bpm.userapp.jndiConnConf" entry on the **Lookup Resource Template** tab and click **Save**.

i. In the **Destination JNDI Name** field, enter the name of the queue to which JMS messages are to be sent. This name must match the queue name in the JMS Destination Configuration. (It is specified in the **userapp.JmsRequest** substitution variable.) In the default configuration, the **userapp.JmsRequest** substitution variable resolves to the queue name, **queue.sample**.

j. On the next dialog, in the **Resource Template** field list, select the JMS Destination Resource Template you just created, then click **Save**.

k. On the Warning dialog, click the **Retry** button.

l. On the next dialog, click the **Re-validate** button.

m. When the message says that the validation was successful, click **Deploy**.

14. On the Administrator **Applications** tab, refresh the list of applications (using the button) until you see "Running" in the **Runtime State** column for the BPM service application.

Chapter 9 ActiveMatrix Decisions Rules Language

This chapter describes the *ActiveMatrix decisions rules language*. This JavaScript-like language can be used in custom actions and conditions in decisions table rules.

Topics

Rule Language Basics

Whitespace

Whitespace is used to separate tokens (identifiers, keywords, literals, separators, and operators) just as it is used in any written language to separate words. Whitespace is also used to format code.

These are whitespace characters, excluding line terminators:

- the ASCII SP character, also known as "space"
- the ASCII HT character, also known as "horizontal tab"
- the ASCII FF character, also known as "form feed"

Line terminators include these characters:

- the ASCII LF character, also known as "newline"
- the ASCII CR character, also known as "return"
- the ASCII CR character followed by the ASCII LF character

Comments

Comment rules as shown:

/* *text* */	Ignores the text from “/*” to “*/”.
// *text*	Ignores the text from “//” to the end of the line.

Separators

The following tokens are used for separators:

;	Statement separator for conditions and actions.
(	Expression grouping begin, or function argument list begin.
)	Expression grouping end or function argument list end.
,	Argument list separator.

Identifier Naming Requirements

An identifier is an unlimited-length sequence of letters and digits, the first of which must be a letter. Letters include uppercase and lowercase ASCII Latin letters A-Z, a-z, and the underscore (_).

Do not use the dollar sign ($).

Identifiers are case sensitive.

Identifiers cannot have spaces (except shared resource identifiers).

Identifiers may not be the same as any literal, keyword, or other reserved word. See Keywords and Other Reserved Words on page 126 and Literals on page 123.

Letters and digits may be drawn from the entire Unicode character set, which supports most writing scripts in use in the world today, including the large sets for Chinese, Japanese, and Korean. This allows programmers to use identifiers in their programs that are written in their native languages.

Digits include the ASCII digits 0-9.

Two identifiers are the same only if they have the same Unicode character for each letter or digit. Note that some letters look the same even though they are different Unicode characters. For example, a representation of the letter A using \u0041 is not the same as a representation of the letter A using \u0391.

Two example identifiers: `new_order` `E72526` `creditCheck`

Here is a more succinct way for programmers to understand the requirements:

```
<Identifier> := [ <ID_START> { <ID_PART> }* ]

<ID_START> := except '$', any character for which java.lang.Character.isJavaIdentifierStart() returns true

<ID_PART> := except '$', any character for which java.lang.Character.isJavaIdentifierPart() returns true
```

Local Variables

You can use local variables of the following types in rule actions and conditions:

- Primitives
- BOM types

You can also use primitive arrays, which are fixed length. Here are examples of array declaration, initialization, and array creation expressions:

- Array declaration and initialization:

```
int i;              // int
int[] ii = {1,2,i};  // array of int
```

- Array creation with initialization expression:

```
ii = int[] {1,2,3};
```

- Array creation without initialization expression:

```
int[] arr = int[5] {};
arr = int[5]{};
```

- Getting the length of the array:

```
int arrLength = arr@length;
```

Literals

The ActiveMatrix decisions rules language supports these literals:

- **int** — One or more digits without a decimal. May be positive or negative. Examples:
  4 45 -321 787878
- **long** — An integer literal suffixed with the letter L. The suffixed L can be either upper or lower case, but keep in mind that the lower case L (l) can be difficult to distinguish from the number one (1).
  Examples: **0l 0777L 0x100000000L 2147483648L 0xC0B0L**
- **double** — A number that can represent fractional values. D suffix is optional unless there is no decimal point or exponent.
  Examples: **1D 1e1 2. .3 0.0D 3.14 1e-9d 1e137**
- **String** — Zero or more characters enclosed in double quotes (""). The string must be on one line. Use \n for newlines. Use the plus sign (+) to concatenate string segments. Examples: empty string: **""** (quotes with no space). Space character: **" "** (quotes with one or more spaces). Strings with values: **"P0QSTN3" "The quick brown fox had quite a feast!"**
  Strings spanning multiple lines:
  "The quick brown fox " +
  "had quite a feast!"
- **boolean** — One of these two values: **true false**
- **Null** — This value: **null**

Escape Sequences

You can represent characters in literals using the escape sequences shown in the following table:

Character	Escape Sequence
backspace	\b
horizontal tab	\t
linefeed	\n
form feed	\f
carriage return	\r
double quote	\"
single quote	\’
backslash	\\

Operators

The language defines the following operators. Operators in this list that are also used in the Java language work the same as the Java operators:

Operator	Notes
++ —	increment, decrement
+ -	unary plus, unary minus
* \ %	multiplication, division, remainder
+ -	addition, subtraction
> < >= <=	greater than, less than, greater than or equal to, less than or equal to
=	assignment
== !=	equality, inequality (Does deep string comparison, unlike Java.)
&& || !	boolean AND, OR, NOT

Operator	Notes
-= += *= /= %=	combined operation and assignment. As an example, *expr* += 2; is the same as *expr* = *expr* + 2; += works on strings as well as numbers. For example, you have a variable String s = "abc"; and then you use s+= "def"; and so the value of s becomes "abcdef"
instanceof	Tests whether an object is an instance of specified type. Example: boolean b = customer instanceof USCustomer;
.	property access
@	attribute access

Keywords and Other Reserved Words

Do not use the words listed in this section as identifiers, resource names, or folder names. Case sensitivity depends on context (as noted in documentation).

$lastmod
abs
abstract
accept
ACTION
AdvisoryEvent
alias
all
and
as
asc
assert
attribute
avg
backwardChain
between
body
boolean
break
by
byte
case
catch
char
class
Concept
CONDITION
const
ContainedConcept
continue
count
date
DateTime
days
dead
declare
default
delete
desc
distinct
do
double
emit
else
entity
enum
Event
except
exists
extends
false
final
finally
first
float
for
forwardChain
from
goto
group
having
hours
if
implements
import
in
instanceof
int
interface
intersect
last
is_defined
is_undefined
key
last
latest
like
limit
lock
long
maintain
max
milliseconds
min
minutes
mod
moveto
native
new
not
null
object
offset
or
order
order
pending_count
package
policy
priority
private
protected
public
purge
QUERY
rank
requeue
return
rule
rulefunction
scope
seconds
select
short
SimpleEvent
sliding
static
strictfp
String
sum
super
switch
synchronized
then
this
throw
throws
time
TimeEvent
timestamp
transient
true
try
tumbling
undefined
union
unique
using
validity
virtual
void
volatile
when
where
while

Attributes

The following attributes can be used in rules to access information of various kinds. Use the @ operator to access attributes.

Entity	Attributes	Type	Returns
ContainedBOM	@parent	BOM	The parent instance. (This is treated as a reference in the language.)
PropertyAtom	@isSet	boolean	True if the property value has been set. Otherwise, false.
PropertyArray	@length	int	The number of PropertyAtom entries in the array.

Working with BOM Properties

This section describes how to access BOM properties using the ActiveMatrix decisions rules language.

BOM Property Atom

This is the syntax for accessing a BOM property atom:

instanceName.propertyName

where *instanceName* is the identifier of the BOM instance, and *propertyName* is the name of the BOM property that you want to access.

For example to get the current value of the cost propertyAtom:

```
int x = instanceA.cost;
```

For example, to set a value with the current system timestamp:

```
instanceA.cost = value;
```

BOM Property Arrays

Accessing a BOM Property Array

This is the syntax for accessing a BOM property array:

instanceName.propertyName

where *instanceName* is the identifier of the BOM instance, and *propertyName* is the name of the BOM property that you want to access.

Accessing a Value in a BOM Property Array

To access a value in a property array, identify the position in the array of the value as shown:

instanceName.propertyName[*indexPosition*]

For example:

```
String x = instanceA.lineItem[0];
```

This gets the current value of the first property atom in the array, lineItem, and assigns it to the local variable, x.

Array index difference In the ActiveMatrix decisions rules language, array indexes start from zero (0). However, in XSLT and XPath languages, they start from one (1). It's important to remember this difference when using the rule language in the rule editor, and when working in the XSLT mapper and the XPath builder.

Setting the Value for an Existing BOM Property Array Position

You can set the value of an existing position in an array. For example:

```
int[] ii = {1,2,3};
ii[2] = 1;
```

Adding a Value to a BOM Property Array

You can append a value to the end of a property array. You cannot, however, add a value to any other position in an array. This is the syntax:

instanceName.propertyName[*indexPosition*] = *value*

To use the syntax shown above you must know the index position of the end of the array. You can append a value to the end of an array without knowing the index position of the end of the array using the @length attribute as shown:

instanceName.propertyName[*instanceName.propertyName*@length] = *value*

Deleting Values in a Property Array

This is the syntax for deleting an array property:

Instance.PropertyArray.delete(*instanceName.propertyName*, *indexPosition*);

Exception Handling

The ActiveMatrix decisions rules language includes try/catch/finally blocks and has an Exception type. The try/catch/finally blocks behave like their same-name Java counterparts.

This section describes the try/catch/finally commands.

Syntax

These combinations are allowed:

- try/catch
- try/finally
- try/catch/finally

try

```
try {
    try_statements
}
```

catch

```
catch (Exception identifier) {
    catch_statements
}
```

finally

```
finally {
    finally_statements
}
```

Flow Control

The ActiveMatrix decisions rules language includes commands to perform conditional branching and iteration loops. This section describes these commands.

if/else

The if/else command allows you to perform different tasks based on conditions.

Syntax:

```
if(condition){
    code_block;
}
else{
    code_block;
}
```

for

The for command allows you to create a loop, executing a code block until the condition you specify is false.

Syntax:

```
for(initialization; continue condition; incrementor){
code_block;
[break;]
[continue;]
}
```

break allows you to break out of the loop.

continue allows you to stop executing the code block but continue the loop.

For example:

```
  for(i=0; i<len; i++){
  total += Finances.assets[i].assetValue;
}
```

while

The while command allows you to perform one or more tasks repeatedly until a given condition becomes false.

Syntax:

```
while(condition){
  code_block;
  [break;]
  [continue;]
}
```

break allows you to break out of the loop.

continue allows you to stop executing the code block but continue the loop.

Rule Language Datatypes

The following subsections provide data type conversion tables, information about operators and types, and information about how the ActiveMatrix decisions rules language handles inconstancy problems with data types.

Compatibility of Operators with Types

The following table shows the compatibility of operators with types.

	Right Side of Operator					
Left Side of Operator	**str**	**int**	**lon**	**dou**	**boo**	**dat**
str	=, +, eq, cmp, inst	+	+	+	+	+
int	+	=, math, eq, cmp	=, math, eq, cmp	=, math, eq, cmp		
lon	+	=, math, eq, cmp	=, math, eq, cmp	=, math, eq, cmp		
dou	+	=, math, eq, cmp	=, math, eq, cmp	=, math, eq, cmp		
boo	+				=, eq	
dat	+,					=, eq, cmp, inst

Abbreviation	Meaning and Notes
boo	Boolean.
cmp	Comparison operators: <, >, <=, >=
dat	Date/Time

Abbreviation	Meaning and Notes
dou	Double
eq	Equality operators: ==, !=
inst	instanceof
int	Integer
lon	Long
math	Numerical operators: unary +, unary -, =, - , *, /, %
str	String

Correcting Inconsistencies of Type

The ActiveMatrix decisions rules language attempts to correct inconsistencies of type whenever possible by converting expressions to the appropriate type. It converts expression types in the following cases:

- An expression uses the plus sign (+) with a string operand.
- An arithmetic expression includes numbers of differing types.
- The value of an expression is assigned to a variable of a different type.
- The value of an expression is passed to a function that declares a different type.

There are some inconsistencies of type that the ActiveMatrix decisions rules language cannot correct. For example, all expressions within conditions must be of type boolean. If an expression within a condition evaluates to anything other than boolean, it would be illogical for the ActiveMatrix decisions rules language to convert the expression to boolean. In cases like this, it returns an error at compile time.

String Operands

When an expression uses the plus sign (+) with a string operand, the ActiveMatrix decisions rules language treats the expression as a request for concatenation rather than addition. It converts the second operand to a string and concatenates the two strings.

For example:

"area code: " + 650 becomes

"area code: 650"

Arithmetic Expressions

The following information applies to these operators:

```
* / % + - < <= > = == !=
```

When an expression uses one of the above arithmetic operators with two numbers of different numeric types, the ActiveMatrix decisions rules language promotes one of the two operands to the numeric type of the other. It makes these promotions as follows:

- If either operand is a double, the ActiveMatrix decisions rules language promotes the other to a double.
- Otherwise, if either operand is a long, it promotes the other to a long.

Assignment Conversion

If the value of an expression is assigned to a variable, the ActiveMatrix decisions rules language converts the expression's type to that of the variable. This might include, for example, converting a double to an int.

Function Argument Conversion

Conversions of function arguments are handled in the same way as assignment conversions.