TIBCO BusinessEvents™

Decision Manager

Software Release 3.0.1
November 2008

The Power to Predict™

TIBCO®
The Power of Now®

Important Information

Contents

Preface

TIBCO BusinessEvents™ allows you to abstract and correlate meaningful business information from the data flowing through your information systems and take appropriate action using business rules. By detecting complex patterns within the real-time flow of simple events, BusinessEvents™ can help you to detect and understand unusual activity, recognize trends, problems, and opportunities. BusinessEvents delivers this business critical information in real time to your critical enterprise systems or custom dashboards. With BusinessEvents you can predict the needs of your customers, make faster decisions, and take faster action.

BusinessEvents
The Power to Predict™

Topics

Enterprise Suite and Inference Edition Features

BusinessEvents is available in the Inference Edition and in the Enterprise Suite. The components available in each option are listed below.

Inference Edition and Enterprise Suite

Inference Edition provides inferencing features and comprises the following components (also included in Enterprise Suite):

- Server—The BusinessEvents runtime engine.
- Workbench—A TIBCO Designer palette of BusinessEvents resources.
- TIBCO ActiveMatrix BusinessWorks 5.x Plug-in—A TIBCO Designer palette of activities that enables communication between BusinessEvents and ActiveMatrix BusinessWorks™. (When you select this option, BusinessEvents Workbench and Server are also automatically selected.)
- Documentation—TIBCO BusinessEvents documentation. The `doc` folder contains an HTML and a PDF folder. If you do not install documentation, this folder is not included in the installation.

Enterprise Suite Only

Enterprise Suite includes all the components in Inference Edition, plus:

- Decision Manager application—A business user rule-building application.

The Decision Manager application is available only on Windows.

- Rules Management Server—A rules server for the Decision Manager application.
- Query—A language and set of functions for querying cache data.
- Database Concepts—A utility for creating concepts from database metadata, with functions for updating the associated database tables or views.
- State Modeler—A component that enables you to model the lifecycle of concept instances.

Related Documentation

This section lists documentation resources you may find useful.

TIBCO BusinessEvents Documentation

- *TIBCO BusinessEvents Installation*: Read this manual for instructions on site preparation and installation.
- *TIBCO BusinessEvents Getting Started*: After the product is installed, use this manual to learn the basics of BusinessEvents. This guide provides step-by-step instructions to implement an example project and also explains the main ideas so you gain both understanding and practical knowledge.
- *TIBCO BusinessEvents User's Guide*: Read this manual for instructions on using TIBCO BusinessEvents to create, manage, and monitor complex event processing projects.
- *TIBCO BusinessEvents Decision Manager*: This manual explains how to use decision tables to create rules using a spreadsheet-like interface, and also how to administer the Rules Management Server.
- *TIBCO BusinessEvents Language Reference*: This manual provides reference and usage information for the BusinessEvents rules language and the BusinessEvents query language.
- *TIBCO BusinessEvents Cache Configuration Guide*: This online reference is available from the HTML documentation interface. It provides configuration details for cache-based object management. Cache-based object management is explained in *TIBCO BusinessEvents User's Guide*.
- *TIBCO BusinessEvents Java API Reference*: This online reference is available from the HTML documentation interface. It provides the Javadoc-based documentation for the BusinessEvents API.
- *TIBCO BusinessEvents Functions Reference:* This online reference is available from the HTML documentation interface. It provides a listing of all functions provided with BusinessEvents, showing the same details as the tool tips available in the TIBCO Designer rule editor interface.
- *TIBCO BusinessEvents Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

You may find it useful to read the documentation for these TIBCO products:

- TIBCO ActiveMatrix BusinessWorks™
- TIBCO Rendezvous®
- TIBCO Enterprise Message Service™
- TIBCO Designer™
- TIBCO Hawk™
- TIBCO Runtime Agent™

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME *BE_HOME*	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco`. Other TIBCO products are installed into an installation environment. Incompatible products and multiple instances of the same product are installed into different installation environments. The directory into which such products are installed is referenced in documentation as *ENV_HOME*. The value of *ENV_HOME* depends on the operating system. For example, on Windows systems the default value is C:\tibco. TIBCO BusinessEvents installs into a version-specific directory within *TIBCO_HOME*. This directory is referenced in documentation as *BE_HOME*. The value of *BE_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\be\3.0`.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand` [**`enable`** | `disable`]

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO BusinessWorks Concepts*. • To introduce new terms. For example: A portal page may contain several *portlets*. Portlets are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *pathname*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
📝	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
💡	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
⚠	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical 'OR' that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand param1 | param2 | param3`

Table 2 Syntax Typographical Conventions

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	
	`MyCommand {param1 param2} | {param3 param4}`
	
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	
	`MyCommand {param1 | param2} {param3 | param4}`
	
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	
	`MyCommand param1 [param2] {param3 | param4}`

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please 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 Introduction to Decision Manager

This chapter introduces the Decision Manager application and Rules Management Server (RMS) along with terminology of the Decision Manager application.

Topics

Introduction to Decision Manager

Decision Manager, a component of TIBCO BusinessEvents, is an Eclipse-based Rich Client Platform (RCP) application. Its friendly user interface allows business personnel with little or no technical background to author, test, and deploy business rules to the BusinessEvents engine. Decision Manager is the client to a server component called RMS, which is a rules management server. RMS manages the lifecycle of decision projects and provides an approval work flow.

- Decision Manager is supported only on Windows. (The server component, RMS, is supported on the same platforms as BusinessEvents.)
- BusinessEvents projects must use Cache object management in order to work with decision tables created in Decision Manager.

Decision Manager simplifies complex business rules by breaking up the logic into multiple simple rules. Each simple business rule is represented by a row in a decision table.

A decision table is an interface with rows and columns for a business user to define threshold values (conditions) and actions in a tabular format. Each row can be thought of as one rule within a table that is made up of many rules.

Terminology note—Rules

In Decision Manager terminology, a *rule* is a business rule that embodies some business logic. It is not the same as a rule in BusinessEvents. In fact, when a decision table is deployed to BusinessEvents, it forms the body of a BusinessEvents *rule function*. Like any rule function it can be called by a BusinessEvents rule, act as an event preprocessor, and so on. To learn about BusinessEvents rules and rule functions, refer to *TIBCO BusinessEvents User's Guide*.

Virtual Rule Functions and Decision Tables

In order to work with Decision Manager decision tables, a BusinessEvents project must contain one or more virtual rule functions. A *virtual rule function* (VRF) is a BusinessEvents rule function with no body, similar to a Java interface. VRFs are brought into Decision Manager (using the project EAR file), where decision tables provide the body (that is, the implementation). The implementation classes are then deployed to the BusinessEvents application.

Rule Building with Decision Manager

Decision Manager enables business users to focus on business logic. It breaks complex rule logic into multiple simpler rules. Here is an example of a fairly complex rule:

Condition

```
criterion.status != "local";
transaction.AccountId == account.Id;
Temporal.Numeric.addAllHistoryDouble(account.Debits,
DateTime.getTimeInMillis(DateTime.addSecond(now(),
scalars.OneDay))) + transaction.Amount > criterion.daily_limit;
```

Action

```
if (criterion.status == "master") {
account.Status = "Suspended";
// other statements
System.debugOut("##### Account id <" + account.Id + " suspended");
} else if (criterion.status == "contender") {
System.debugOut("##### Account id <" + account.Id + " suggest
suspension");
} else {
System.debugOut("##### Unknown criterion");
// other statements
}
```

In Decision Manager, conditions are presented in a more straightforward manner, for example:

```
Person.age < Max(20, Parent.age)
Person.creditscore >= Math.function(...)
Person.gender == "female"
```

The rule action can also be specified in a straightforward way, such as the following:

```
Application.status = "ACCEPTED"
Application.credit = 4000
sendNotification()
```

This simplified set of conditions and actions can be easily modeled in a decision table.

RMS and Decision Projects

Using the rules management server (RMS) a more technical user, generally referred to as a *rule administrator*, builds RMS projects. RMS is also used by technical users to check and approve (or reject) decision tables submitted by business users before they are deployed.

An RMS project consists of the following:

- The BusinessEvents project's ontology model, which is a set of concepts, scorecards, and events that represent the objects and activities in your business. BusinessEvents projects are created using TIBCO Designer

- Decision tables
- Domain model (if needed)
- Test data (as desired)
- Access control files

A less technical user, a role referred to as a *business user*, checks out an *RMS project* and saves it locally as a *decision project*. The user works with the decision project—saving, modifying, validating, testing, and so on—before committing it to RMS for approval. See Introduction to Rules Management Server on page 5 for more information.

Introduction to Rules Management Server

You can configure RMS extensively. The product documentation describes the product as shipped, for example, it uses the workflow status values and general workflow behavior. If your installation has been customized, it may not exactly match product documentation. For example, you can implement a multi-stage approval, rather than the one-step approval process shipped with the product.

RMS is a lightweight rules management server component for managing the rules management repository. RMS provides user authentication features, decision project authorization and other project management features, and a project approval workflow.

Decision Manager communicates with RMS to retrieve decision projects, update local copies of decision tables, and commit changes to the decision table repository. RMS users approve or reject such changes.

- RMS is supported on all platforms that support BusinessEvents.
- Users can access RMS remotely. It does not have to be installed on users' machines.

RMS is implemented using BusinessEvents, and knowledgeable BusinessEvents users can customize it. The BusinessEvents project for RMS contains a state machine for an approval process that can be customized, for example.The BusinessEvents project is located in the *BE_HOME*`/rms/project` directory.

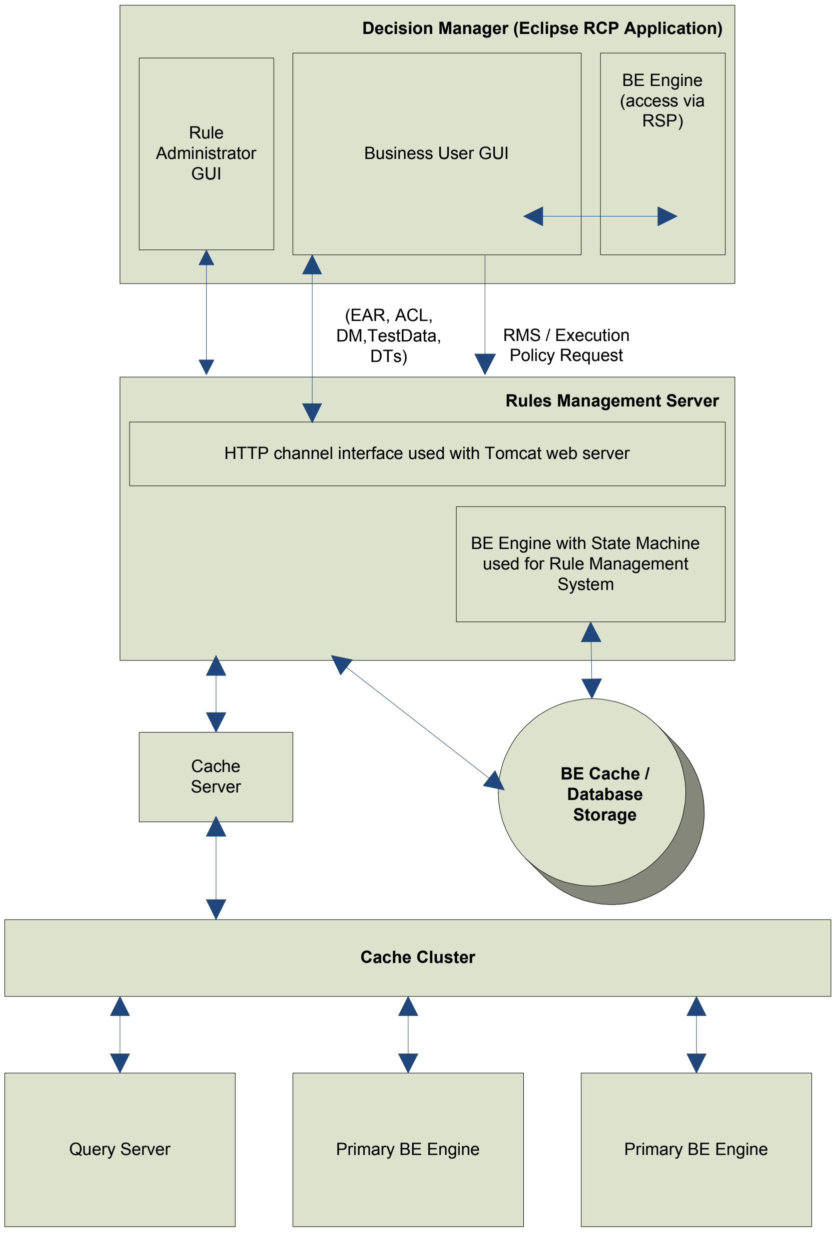
RMS provides an easy, secure, and scripted deployment lifecycle. It enables users to do the following:

- Set up projects with optional domain models (which define acceptable values for a property) and optional predefined test data
- Set project security policies
- Approve or reject committed projects and make comments
- Check on the status of all committed implementations or projects and keep track of all project versions
- Deploy approved implementations or projects to a running TIBCO BusinessEvents engine, and also undeploy implementations.

When Decision Manager users check out projects, a copy of the project is copied to their local machine so they can build decision tables. When the tables are committed to the server for approval, an RMS server user decides if the tables are ready for deployment. Class files are built and deployed to the runtime BusinessEvents application.

Deployment Architecture

This figure illustrates a typical deployment of the TIBCO BusinessEvents Decision Manager and RMS:


Decision Manager (Eclipse RCP Application)
Rule Administrator GUI
Business User GUI
BE Engine (access via RSP)
(EAR, ACL, DM,TestData, DTs)
RMS / Execution Policy Request
Rules Management Server
HTTP channel interface used with Tomcat web server
BE Engine with State Machine used for Rule Management System
Cache Server
BE Cache / Database Storage
Cache Cluster
Query Server
Primary BE Engine
Primary BE Engine

Decision Manager and RMS Workflow

This section describes how an RMS project is created and used by Decision Manager users. This section assumes initial RMS configuration has been done. It describes the approval workflow shipped with the product.

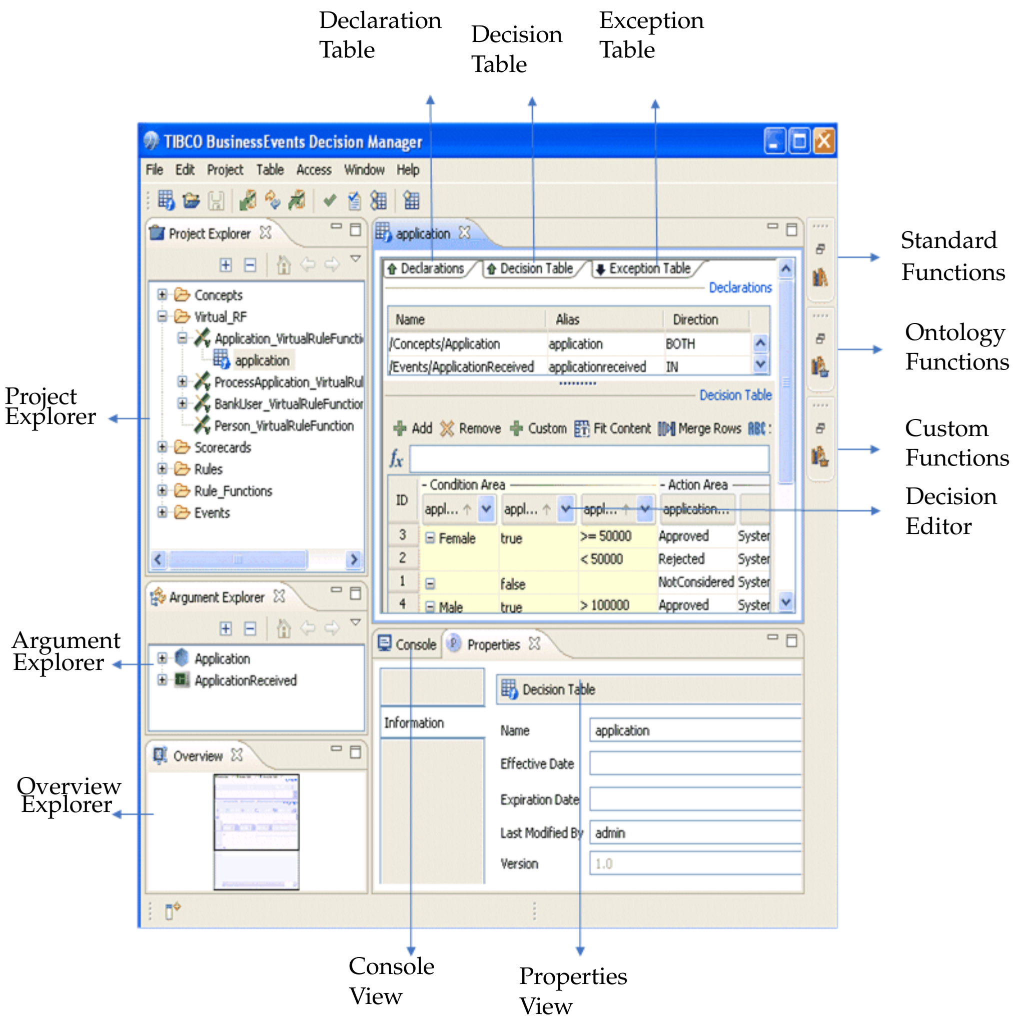
RMS can be customized. Different user roles can be given different permissions, and the approval workflow can be changed and extended. The following is a general overview of the procedures in the product as shipped.

1. A BusinessEvents user creates a TIBCO BusinessEvents project, creating the ontology, and writing rules that make use of virtual rule functions (VRFs). (VRFs are implemented later by a business user using decision tables.)
2. An RMS user takes the EAR file from that BusinessEvents project and sets up an RMS project for it. He or she creates access control files, and other files that are needed by the project such as domain model files and test files.
3. An RMS user starts the RMS server.
4. A business user logs in to Decision Manager and requests a decision project by checking it out and saving the project to disk.
5. The business user creates one or more decision tables and saves the modified decision project locally. Business users with permission can create domain models to make data entry more reliable.
6. The business user tests the decision tables locally with a locally running BusinessEvents engine started by Decision Manager automatically. Business users with permission can create test data.
7. When satisfied, the business user commits the decision table for approval.
8. An RMS user receives the request and reviews the project and then approves or rejects it.
9. RMS generates class files in a known location on a production BusinessEvents engine. (Or the files can be manually copied there as needed.)
10. The class files are either hot deployed to a running system, or are deployed when the BusinessEvents system starts up, depending on configuration and timing.

The end result is that the business logic defined by the business user is now part of the BusinessEvents application.

Decision Manager User Interface

This figure shows the main page with a decision project loaded and a particular decision table opened:



Name	Alias	Direction
/Concepts/Application	application	BOTH
/Events/ApplicationReceived	applicationreceived	IN

ID	appl...	appl...	appl...	application...	
3	Female	true	>= 50000	Approved	Syster
2			< 50000	Rejected	Syster
1		false		NotConsidered	Syster
4	Male	true	> 100000	Approved	Syster

Decision Table

Name	application
Effective Date	
Expiration Date	
Last Modified By	admin
Version	1.0

Project Explorer

The Project Explorer displays the BusinessEvents ontology. This ontology is used by the business user to work with a decision table. This is the ontology that was created in TIBCO Designer where all the models and rules were created.

Argument Explorer

Argument Explorer displays the arguments of the VRFs which are defined in TIBCO Designer. The business user can expand the entities (concepts or events) to see their properties, decide on which properties to use to make a decision, and drag and drop the properties onto the Condition or Action area of the Decision Table editor to create an implementation. Primitives supported by BusinessEvents can also be dragged.

Overview Explorer

This is an outline view for the decision table editor. It can be used to navigate larger tables.

Console View

All engine messages are logged in this panel. This panel displays information about errors resulting from abnormal execution of the application.

Problems View

All BusinessEvents validation errors are shown in this panel. The types of errors that may be displayed upon validation include language validation errors, access control errors, and so on. This view does not display information about errors resulting from abnormal execution of the application.

History View

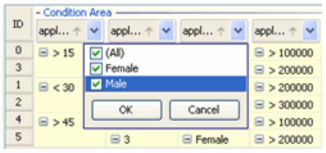
This panel displays a list of all the check-ins made for a decision table (similar to the revisions list or the log feature provided by most source control clients).

Properties View

This panel displays the metadata associated with the various entities (concepts, events, rules, and so on) that compose a BusinessEvents project. This view is activated when you click on a project entity (concept, event, rule, and so on) that has property data associated with it. From the Property Explorer you can modify the effective date and expiration date of decision table entities only.

Domain Model Editor

The domain model editor enables authorized users to define and modify sets of valid values for ontology properties. Users setting up a decision table can then select from the allowable values when using properties to define a decision table row. For example, the following figure shows how a domain model is used to select a value for a gender property:



ID	appl...	appl...	appl...	appl...
0	> 15			> 100000
3				> 200000
1	< 30			> 200000
2				> 300000
4	> 45			> 100000
5		3	Female	> 200000

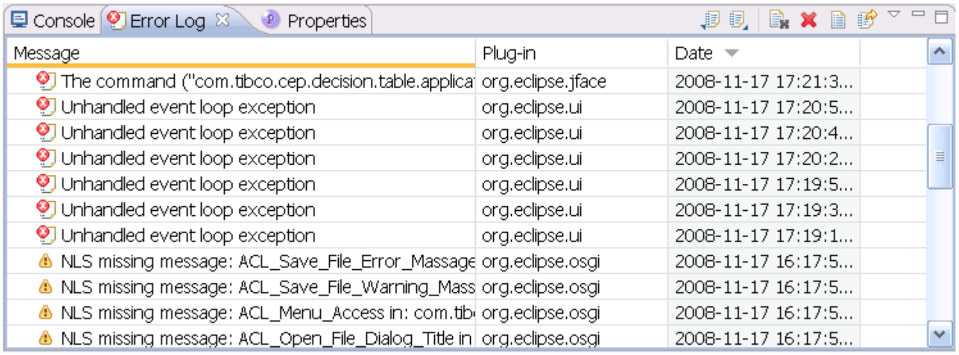
Decision Table Editor

This is the work area for defining the decision tables. The editor consists of three different sections:

- **Declaration Table** Displays read-only arguments that are defined in TIBCO Designer for the implementation.
- **Decision Table** The table used to define business rules by dragging and dropping properties of the arguments.
- **Exception Table** Another instance of a decision table where you enter the conditions and actions for exceptions. For example, you could add rows that capture situations where fields are blank or contain invalid values. In such cases, you can add custom actions to send notifications or set return values in your table accordingly.

Error Log View

Shows a list of errors, with convenience controls such as export, import, clear log viewer, delete log, open log, and restore log. You can double click on any item to display a detailed view.



Message	Plug-in	Date
The command ("com.tibco.cep.decision.table.applica	org.eclipse.jface	2008-11-17 17:21:3...
Unhandled event loop exception	org.eclipse.ui	2008-11-17 17:20:5...
Unhandled event loop exception	org.eclipse.ui	2008-11-17 17:20:4...
Unhandled event loop exception	org.eclipse.ui	2008-11-17 17:20:2...
Unhandled event loop exception	org.eclipse.ui	2008-11-17 17:19:5...
Unhandled event loop exception	org.eclipse.ui	2008-11-17 17:19:3...
Unhandled event loop exception	org.eclipse.ui	2008-11-17 17:19:1...
NLS missing message: ACL_Save_File_Error_Massage	org.eclipse.osgi	2008-11-17 16:17:5...
NLS missing message: ACL_Save_File_Warning_Mass	org.eclipse.osgi	2008-11-17 16:17:5...
NLS missing message: ACL_Menu_Access in: com.tib	org.eclipse.osgi	2008-11-17 16:17:5...
NLS missing message: ACL_Open_File_Dialog_Title in	org.eclipse.osgi	2008-11-17 16:17:5...

Error Log View makes visible this log file:

BE_HOME`\DecisionManager\workspace\.metadata\.log`

This view is not visible by default. To enable it, set the property `bui.show.ErrorLog=true` in the `bui-config.tra` file. See Configuring Decision Manager Properties on page 21.

Catalog Functions

There are three catalogs of functions:

- Standard Functions
- Ontology Functions
- Custom Functions

Expand each set to view the functions. You can drag and drop functions onto the desired decision table cell to define a condition or action. When you float the cursor over a function in the registry, Decision Manager displays a tooltip showing the description and syntax.

Custom Functions Advanced users can create custom functions (with their own tool tips). See Adding Custom Functions to Decision Manager on page 34.

To view all keyboard shortcuts in Decision Manager select Help > Key Assist.

Chapter 2 RMS and Decision Manager Configuration

This chapter explains the directory structure of the Rules Management Server (RMS) server and for each RMS project. It also explains the configuration tasks for RMS, for RMS projects, and for Decision Manager.

Ensure you have done any post-installation configuration required. For example on UNIX platforms, user permissions must be given to certain directories. See *TIBCO BusinessEvents Installation*.

Topics

RMS Project Directories and Files

RMS requires certain files and folders at the server level. These are used by RMS projects. RMS requires project-level files and folders to be structured according to various RMS configuration properties. This section shows the default structures, in the product as shipped.

RMS projects use various files that use required names and must be placed in files and directories with required names within the *project directory*. (See Choosing Single Project or Multiple Project Mode for more on the location of the project directory). Some project directory names are configurable in the `be-rms.tra`.

Single and Multi-Project Mode Directory Structure

In multi-project mode All project directories are located under a base location directory. The base location directory as shipped contains two configuration files that apply to all projects.

```
users.pwd
WorkflowStages.xml
```

The users.pwd file is used only with file-based authentication.

Choosing a mode and defining locations for project resources is done in the `be-rms.tra` file. The following properties define the multi-project mode base location and the location of files within it:

```
java.property.rms.projects.baselocation
rms.auth.file.location
java.property.rms.project.workflow.config.file
```

In single project mode The `users.pwd` and `WorkflowStages.xml` files are located in the project directory itself.

In both modes, each RMS project directory contains the following subdirectories:

Table 3 RMS Project Directories

Directory	Notes
`/bin`	Contains the EAR file for the decision project.
`/client`	Not used in this release.
`/config`	Contains project configuration files: • The access control file, *RMSProjectName*`.ac`, • The domain model file (as needed) *ProjectName*`.dm`,

Table 3 RMS Project Directories (Cont'd)

Directory	Notes
`/data`	Contains test data in subdirectories, beginning with the `/data/`*ProjectName*`_TestData` directory.
`/decisiondata`	Directory to store approved VRF implementations. Default is `decisiondata`. This directory name is configurable using the property `rms.project.decisiondata`
`/deployment`	Contains the generated class files for the decision tables. This directory name is configurable. `rms.project.deployment`
`/src`	Not used in this release. Can be used to store the TIBCO Designer project source.
`workspace`	A storage area which contains decision project folders and files submitted for approval by Decision Manager users, along with the with the name of the user who submitted it, the version number, and a status = `Approval Pending`. This directory name is configurable using the property `rms.project.workspace`

Configuring RMS Server Properties

RMS works out of the box, but you may wish to make some changes to the server configuration. You only have to change startup properties, host, port, and location of the repository, if the server location or repository location changes. However, you may want to change the location under which project folders are kept.

After you have completed basic configuration you can create RMS projects. See Creating a Rules Management Server Project on page 44 for the procedure and links to related activities.

Choosing Single Project or Multiple Project Mode

You must choose one of the supported modes: Single project or multiple project mode. You can't use both modes. By default multi-project mode is enabled.

With single-project mode, one RMS manages the life cycle of a single project. This model is useful for a deployment scenario where disjunct RMS servers operating from different locations are designated to manage one BusinessEvents project each.

Single Project Mode Project Directory

To set the project directory and project name, for single project mode, use these properties:

```
java.property.rms.project.location
java.property.rms.project.name
```

(And comment out the `java.property.rms.projects.baselocation` property.)

Multiple Project Mode Base Location Directory

With multi-project mode, one RMS manages the life cycle of multiple projects, which are uploaded to a location known to the server. You set a base project directory using this property:

```
java.property.rms.projects.baselocation
```

(And comment out the single-project mode properties.) Project directories are created within the specified base location directory. See RMS Project Directories and Files on page 14.

Enabling Remote Connectivity

As shipped, the `tibco.clientVar.RMS/hostname` property is set to `localhost`. This setting enables the product to be used in non-production settings. For production settings take the following actions:

- Ensure that the hostname property is set to the RMS server host machine name or IP address.
- Ensure that Decision Manager clients have direct network connectivity to the RMS server.

Connections from clients may not work if the RMS server is behind a firewall.

To Configure RMS Properties

See Table 4, RMS Server Configuration Properties for information about each property.

1. Open the *BE_HOME*`/rms/bin/be-rms.tra` file.
2. Change the properties as needed and save the file.

RMS Server Configuration Property Reference

See also Authentication Property Reference on page 41

Table 4 RMS Server Configuration Properties

Property	Notes
`rms.projects.baselocation`	Specifies the location of a directory which contains one or more RMS projects. This property is used in multi-project mode. This is the base location for many other paths defined in the `tra` file when multiple-project mode is used. If you are using multi-project mode, comment out `rms.project.location` and `rms.project.name` or do not provide a value for those properties. Value as shipped is `C:/tibco/be/3.0/rms/examples`.
`rms.project.location`	Specifies the absolute path of the directory which contains the RMS project specified by the property `rms.project.name`. This property is used in single-project mode. If you are using single-project mode, comment out `rms.projects.baselocation` or do not provide a value for that property.
`rms.project.name`	Specifies the name of the project to be managed. This property is used in single-project mode. The name of the project is the name of the project root directory. For example: `CreditCardApplication`. If you are using single-project mode, comment out `rms.projects.baselocation` or do not provide a value for that property.
`tibco.clientVar.RMS/storageLocation`	Specifies the absolute path to create a repository for RMS to store information about all data interactions with Decision Manager. The repository serves as a persistent store for all concepts or events created in the RMS engine using a Berkeley database as persistence mechanism. Default location is `C:/bdb`

Table 4 RMS Server Configuration Properties (Cont'd)

Property	Notes
`rms.project.workspace`	Directory to store checked-in projects. Default is `workspace`. This directory is relative to the project directory.
`rms.project.decisiondata`	Directory to store approved implementations. Default is `decisiondata`. This directory is relative to the project directory.
`rms.project.deployment`	Directory to store deployable class files. Default is `deployment`. This directory is relative to the project directory. See Chapter 9, Deploying and Unloading Decision Tables, on page 101 for more details. **Tip**: If you configure the corresponding BusinessEvents property `be.engine.cluster.externalClasses.path` to point to the same directory, then when decision tables are approved, they are automatically available for deployment to BusinessEvents without manual copying. See Virtual Rule Functions and Decision Manager in *TIBCO BusinessEvents User's Guide* for details.
`java.property.rms.auth.file.location`	Specifies the name of the password file. By default the filename is `users.pwd`. You can specify a subdirectory in addition to a filename. Place this file (and optional subdirectory path) in the appropriate location: Single-project mode: `rms.project.location/rms.project.name/config` Multiple-project mode: `rms.projects.baselocation`. See Chapter 3, Configuring User Authentication, on page 37 for more information on this and other authentication-related properties.
`java.property.rms.project.workflow.config.file`	Specifies the name of the workflow permissions file. By default the filename is `WorkflowStages.xml`. You can specify a subdirectory in addition to a filename. RMS reads this file (and optional subdirectory path) in the following locations: For single-project mode, in the `rms.project.location directory.` For multiple-project mode, in the `rms.projects.baselocation` directory.

Table 4 RMS Server Configuration Properties (Cont'd)

Property	Notes
`tibco.clientVar.RMS/hostname`	Specifies the host name or IP address of the machine where RMS is hosted so that remote clients can connect to the server. See Enabling Remote Connectivity on page 17 for details. Default is `localhost`.
`tibco.clientVar.RMS/port`	Specifies the port number of the machine where RMS is listening to the client request. Default is `5000`.

Configuring Decision Manager Properties

Decision Manager is configured to work with RMS as shipped. However, for various reasons, you may want to alter the default configuration.

Restart Decision Manager after making any changes to the `bui-config.tra` file.

To Set Decision Manager Configuration Parameters

See Table 5, Decision Manager Configuration Properties for information about each property.

1. Open the *BE_HOME*`/DecisionManager/configuration/bui-config.tra`
2. Change the properties as needed and save the file.

Decision Manager Configuration Property Reference

Below are properties you can set in the `bui-config.tra` file.

Table 5 Decision Manager Configuration Properties

Property	Notes
`rms.host`	Specifies the host name of the machine where RMS is running. Uses the same host name with which RMS is started. For example, if RMS is started with hostname `host.company.com`, the same name, `host.company.com` must be used in the `rms.host` property. (See the relevant properties in *BE_HOME*`/rms/bin/be-rms.tra`.) Default is `localhost`.
`rms.port`	Specifies the port number for the RMS server. Default is `5000`.
`rms.context.root`	Specifies the URL on which RMS is listening to requests over HTTP transport. (In BusinessEvents terms, this is the channel on which the BusinessEvents RMS application listens.) Default is `/Transports/Channels/HTTPChannel/`
`tibco.bui.codegen.prepend_classpath`	References all classes needed for testing. Used for BusinessEvents-ActiveMatrix BusinessWorks integration projects. See Configuring Project Tester Options on page 31.
`Engine.Log.Dir`	Specifies the location of the engine log file. Default is `../logs`
`be.trace.log.enable`	Enables tracing to log files.
`be.trace.enable`	Enables tracing features. All other trace properties are ignored if `be.trace.enable` is set to false.
`be.trace.term.enable`	Enables tracing to terminal.
`be.trace.term.sysout.redirect`	Redirects the system output to debug logger Default value is true

Table 5 Decision Manager Configuration Properties (Cont'd)

Property	Notes
`be.trace.roles`	Comma-separated list of trace roles to enable. Allowed values: `infoRole`, `errorRole`, `warnRole`, `userRole`, `debugRole` Default value is `infoRole, errorRole, warnRole, userRole`
`bui.demand.load.table`	If set to true, Decision Manager lazily loads the decision tables in the decision project. This is recommended for projects with several large decision tables. It reduces memory consumption for tables that are not opened.
`bui.extended.classpath`	If the `bui.tester.engine.feature.level` property is set to `local` or `full`, then all directories and JAR files are processed and all JAR files are added to the Decision Manager classpath. For example, if the decision project makes use of channels you might set the path as follows: `bui.extended.classpath=%BE_HOME%/lib;%BE_HOME%/hotfix/lib;%TPCL_HOME%/tomcat/server/lib/jakarta-regexp-1.2.jar` Also see notes for `bui.tester.engine.feature.level`.
`bui.gen.class.option`	A value of true makes a context menu option called Generate Class appear in the decision table context menu (see Generate Class Option on page 75). The Generate Class option enables the user to generate the selected decision table's classes. These can be used for testing purposes.
`bui.log.fileName`	If set, logs `output` to the given file.
`bui.show.ErrorLog`	A value of `true` makes the `Error log view` visible in Decision Manager. See Error Log View on page 10
`bui.redirect.httplog`	A value of `true` redirects `http log calls` to the Decision Manager Console view.
`bui.redirect.stderr`	A value of `true` redirects `stderr` to the Decision Manager Console view.
`bui.redirect.stdout`	A value of `true` redirects `stdout` to the Decision Manager Console view.

Table 5 Decision Manager Configuration Properties (Cont'd)

Property	Notes
`bui.tester.engine.feature.level`	Specifies settings for testing decision projects. Possible values are `minimal`, `local`, and `full`. See Configuring Project Tester Options on page 31 for details. Also see Testing Decision Tables on page 81. Default is `minimal`.

Configuring Workflow Status Permissions for User Roles

RMS incorporates a workflow for approval of decision artifacts created through Decision Manager. Each stage in the approval process has an associated status.

The approval workflow is modeled as a BusinessEvents state machine which is configured in the BRMS project, which is the BusinessEvents project that provides RMS functionality.

You can customize the default workflow by customizing its state machine in the BRMS BusinessEvents project. This is an advanced activity that assumes advanced knowledge of BusinessEvents. It is not documented in this guide.

Reviewers with different roles may need to review a decision artifact before it becomes available for deployment. You can assign each role an appropriate set of status values. To do this you configure the workflow stages file. For example:

```
<stages>
  <role name = "RULE_ADMINISTRATOR">
     <statuses>
        <status>Approve</status>
        <status>Reject</status>
        <status>Unload</status>
     </statuses>
  </role>
</stages>
```

Users see only the statuses that their role permits them to see.

You must use roles that have been configured for the system. See User Authentication Overview on page 38 for details.

To Configure Workflow Permissions

1. Open the `WorkflowStages.xml` file. As shipped, the file is located in *BE_HOME*`/rms/examples/WorkflowStages.xml`.

   The name and location are configurable, as described in the notes for the `java.property.rms.project.workflow.config.file` property. See Table 4, RMS Server Configuration Properties, on page 18.

2. Within the section for each role name, add, delete, or modify `status` elements so that the set of statuses listed for each role matches the set of statuses you want that user role to be able to use.

3. Save the file.

Configuring for Deployment of Decision Table Classes

After a decision project is approved, class files for the decision tables are generated in the `deployment` subdirectory of the RMS project directory. You can selectively deploy them to the production system.

See Deploying Decision Tables on page 102 for the related administration procedures.

You can also undeploy class files that were earlier deployed. See Configuring for Unloading of Decision Table Classes on page 29).

Requires Cache Object Management Deployment and undeployment of decision table classes requires Cache object management to be used in the BusinessEvents application.

Class Loaders

When classes are being deployed, a BusinessEvents node that has been configured as a class loader loads the generated classes to all the other nodes.

Nodes that contain only query agents cannot function as class loaders. Choose cache server nodes or nodes containing inference agents (or both).

Hot Deployment

You can also configure for hot deployment to a running system. Hot deployment (and one method of undeployment of classes) uses JMX. JMX is a Java standard for managing and monitoring Java applications and services. All agents expose `MBeans` that can be used from the JConsole application. An `MBean` is a managed bean, which is a Java object that can represent a manageable resource. For an introduction to JMX, see the following:

http://java.sun.com/developer/technicalArticles/J2SE/jmx.html

To Configure for Deployment of Decision Table Classes

1. Open the *BE_HOME*`/bin/be-engine.tra` file for the node or nodes you will use for loading the classes.

For deployment at startup

2. Configure the following properties in each node that is to act as a class loader:

   `be.engine.cluster.externalClasses.path` *filepath-to-RMS-classes*

   `be.engine.cluster.externalClasses.classLoader=true`

For hot deployment

3. For hot deployment, ensure that the following entry is present in the `java.extended.properties` property in **all** BusinessEvents nodes:

   `-javaagent:%BE_HOME%/lib/cep-instrumentation.jar`

   The above property is required for hot deployment and is present in the TRA file as shipped.

   See Table 6, BusinessEvents Engine Property Settings for Deploying Decision Table Classes for details on property settings.

4. For hot deployment, also configure the following JMX-related properties in **all** BusinessEvents nodes. Some typical values are shown. Configure for your environment as needed.

```
# Coherence JMX properties

java.property.com.sun.management.jmxremote=true
java.property.com.sun.management.jmxremote.ssl=false
java.property.com.sun.management.jmxremote.port=5558
java.property.com.sun.management.jmxremote.authenticate=false
```

Table 6 BusinessEvents Engine Property Settings for Deploying Decision Table Classes

Property	Notes
`be.engine.cluster.externalClasses.path`	Specifies the filepath used by the cluster to load external rule classes created in Decision Manager to all BusinessEvents cluster nodes. Set on one or more nodes as needed. Set this property in nodes where you also set `be.engine.cluster.externalClasses.classLoader` to true. Ensure these nodes can access the filepath. **Tip**: If the value of this property is set to the value of the RMS property `rms.project.deployment` then when decision tables are approved, they are automatically available for deployment to BusinessEvents without manual copying.

Table 6 BusinessEvents Engine Property Settings for Deploying Decision Table Classes (Cont'd)

Property	Notes
`be.engine.cluster.externalClasses.classLoader`	If true, this node has the ability to load external rule classes to all cluster nodes. Set on one or more nodes as needed. For related details, see notes for `be.engine.cluster.externalClasses.path`. **Note**: Do not use nodes that contain only query agents. Possible values: `true` or `false`. Default value is `true`.
`java.property.com.sun.management.jmxremote.ssl`	Enables secure monitoring via SSL. If `false`, then SSL is not used. Possible values: `true` or `false`.
`java.property.com.sun.management.jmxremote.port`	Enables monitoring and management from remote systems on the specified port. Specify an unused port.
`java.property.com.sun.management.jmxremote.authenticate`	If set to `false`, JMX does not use passwords or access files. Access is available to all users. Possible values: `true` or `false`.

Configuring for Unloading of Decision Table Classes

Users with correct permissions can unload (undeploy) previously deployed decision tables from a running BusinessEvents system.

The procedure for performing the unload using Decision Manager Worklist view is provided in the section Undeploying (Unloading) Decision Tables on page 105. Before you can perform the procedure, however, you must perform configuration described in this section.

Two Ways to Unload Decision Table Classes

You can unload decision tables in two ways:

This feature uses RMI (Java Remote Method Invocation).

- **Using a JMX-based unload operation** If you have configured JMX to hot-deploy classes, you can also unload classes using JMX, with no additional configuration. See Configuring for Unloading of Decision Table Classes on page 29.
- **By selecting the Unload action in the Worklist view** This is a remote operation and requires RMI settings to be configured as explained below.

When you select the Unload action in the Worklist view, Decision Manager sends the request to RMS, which relays it to the RMI server in the running BusinessEvents system, and the class (that is the decision table) is unloaded.

The BusinessEvents system must be running in order for the unload to work.

About RMI

RMI is a Java standard that enables remote communication between Java programs. For an introduction to RMI, see the following:

http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

Configuring to Enable Unload in the Worklist View

You must configure the RMI properties in the following files:

- The *BE_HOME*`/rms/bin/be-rms.tra` file.
- The *BE_HOME*`/be/bin/be-engine.tra` file in all BusinessEvents nodes (except those running only query agents).

1. In each file listed above, add or uncomment and configure the RMI properties as needed. For example:

   ```
   be.engine.cluster.rmi.enabled=true
   be.engine.cluster.rmi.host=localhost
   be.engine.cluster.rmi.port=9999
   ```

2. Save the file.

Table 7 RMS Property Settings for Undeploying Decision Table Classes

Property	Notes
`be.engine.cluster.rmi.enabled`	Specifies if RMI functionality is enabled or not. Set to true to enable. Default is false
`be.engine.cluster.rmi.host`	Specifies the host name of the RMI host, that is, the name or IP address of a machine hosting one or more BusinessEvents nodes. An RMI server will be started on this machine. **Note**: Do not use nodes that contain only query agents. Default is `localhost`.
`be.engine.cluster.rmi.port`	Specifies the port to use when starting the RMI server. If a machine has two BusinessEvents nodes, and you want to start two RMI servers, one for each node, specify a different port for each node. Default is 9999.

Configuring Project Tester Options

When you test a decision project in Decision Manager, you can run the project in a restricted manner, or you can configure to more closely approximate the production system in action. Certain options require project-specific configuration. This section describes the options and explains how to configure for each option.

Project Tester Options

This section describes the options. Each option is selected using the property `bui.tester.engine.feature.level` (see Configuring Decision Manager Properties on page 21).

You must use `local` or `full` to test projects for which entries are added to the classpath.

For example, use `local` or `full` if the project uses custom functions, or is a BusinessEvents-ActiveMatrix BusinessWorks integration project running with a BusinessEvents container. (Decision Manager does not work with integration projects where ActiveMatrix BusinessWorks is the container.)

Minimal

- Does not start BusinessEvents channels at startup.
- Disregards BusinessEvents project configuration and uses in-memory object management.
- The `bui.extended.classpath` property is not processed, and any JAR files listed are not available to the test engine.

This is the default option, and requires no additional configuration. If the project does not run successfully with this option, try using the `local` option.

Local

- Does not start BusinessEvents channels at startup.
- Disregards BusinessEvents project configuration and uses in-memory object management.
- Makes all classes in the `bui.extended.classpath` available to the test engine.

Most projects can run in this mode. It can be slower to run than the `minimal` option. Configuration may be required to make properties set in the BusinessEvents property file, `be-engine.tra`, available to the tester. See Configuring for Local or Full Tester Options on page 32.

Full (Expert mode)

- Starts BusinessEvents channels at startup.
- Uses the BusinessEvents project's object management configuration.
- Makes all classes in the `bui.extended.classpath` available to the test engine.

This option runs the project in the tester engine as closely as possible to the way it runs in the BusinessEvents engine.

Configuration is required to make properties set in the BusinessEvents property file, `be-engine.tra`, available to the tester. Only expert users should attempt this mode. See Configuring for Local or Full Tester Options.

Configuring for Local or Full Tester Options

To configure for the local or full tester options, various properties set in the `be-engine.tra` file must also be added to the `bui-config.tra` file or `DecisionManager.ini` file, as explained in this section.

For the `local` option, you only need to define the `bui.extended.classpath` (see Classpath Configuration). For the full option you need to add more properties, to more fully reflect the configuration in the `be.engine.tra` file.

Where and how you add a property from the `be-engine.tra` file depends on whether it is a BusinessEvents engine property or not:

- Copy BusinessEvents engine properties to the `bui-config.tra` file. (For example, `be.engine.cluster.hasBackingStore`, is an engine property.)
- Add all other required properties as `vmargs` in the `DecisionManager.ini` file:
  - — Prepend the name with `-D`.
  - — If the property begins with `java.property`, also remove `java.property`. For example `java.property.tangosol.coherence.cluster` becomes `-Dtangosol.coherence.cluster`.

Classpath Configuration

If you have to add entries to the classpath of the tester engine such as custom function JAR files, add them to the `bui.extended.classpath` property in the bui-config.tra file (see Configuring Decision Manager Properties on page 21).

Also ensure that all environment variables in the bui.extended.classpath are defined as vmargs in the DecisionManager.ini file.

As explained earlier in this section, you add all non-engine environment variables to the DecisionManager.ini file. For example, when configuring to test integrated decision projects where BusinessEvents is the container, you may need to provide ActiveMatrix BusinessWorks classpath information. Suppose a such a project requires this property in the be-engine.tra file:

```
java.property.palettePath=%BW_HOME%/lib/palettes%PSP%%BW_PLUGINS_H
OME%/lib/palettes%PSP%%TRA_HOME%/lib/palettes
```

You would add the information to the DecisionManager.ini file as follows:

```
-DBW_HOME=c:/tibco/bw/5.4
-DBW_plugins_home=c:/tibco/bw/plugins
-DpalettePath=%BW_HOME%/lib/palettes;%BW_PLUGINS_HOME%/lib/palette
s;%TRA_HOME%/lib/palettes
```

In BusinessEvents-ActiveMatrix BusinessWorks in process integration projects, the container application manages state and other aspects of the project.

You can use Decision Manager with such integration projects only where BusinessEvents is the container. When ActiveMatrix BusinessWorks is the container, BusinessEvents can’t use cache object management, which is required in order to work with decision tables from Decision Manager.

This integration feature is documented in Chapter 12, In-Process ActiveMatrix BusinessWorks Integration, in *TIBCO BusinessEvents User’s Guide*.

Adding Custom Functions to Decision Manager

To configure Decision Manager to use custom functions, you must complete the tasks below.

You can add multiple custom function JAR files (see Task D) but only functions provided in the JAR file specified in the `FUNC_JAR` property (see Task B), display in the Custom Functions view. (You can use functions from additional custom function JAR files by entering them manually.)

Task A Create the Functions and Place them in the Class Path

1. Create the custom function or functions. See *TIBCO BusinessEvents Language Reference* for guidelines.
2. Place the JAR file containing the custom function or functions in a directory in the code generation classpath.

Task B Add the Custom Function JAR Path to the Initialization File

The custom functions provided in the JAR specified by the FUNC_JAR property appear in the custom functions view in the Decision Manager UI. You can specify only one JAR file in this property.

1. Open the *BE_HOME*`/DecisionManager/DecisionManager.ini` file.
2. Add the `FUNC_JAR` property and provide the fully-qualified path to the JAR file. This property takes only one JAR file as its value. For example:

   ```
   DFUNC_JAR=C:/tibco/custom/customfunctions.jar
   ```

3. Save the file.

Task C (For Multiple JARs only) Update the Decision Manager Extended Classpath and Set Tester Level

If you have only one custom function JAR file, skip this step. (The property FUNC_JAR performs this task, but only for one JAR.) If you have multiple custom function JARs do the following:

1. Open the *BE_HOME*`/DecisionManager/configuration/bui-config.tra` file.
2. Add the fully qualified path to the custom functions JAR files in the `bui.extended.classpath` property. For example:

```
bui.extended.classpath=C:/custom/customfunctions.jar;
c:/custom/morefunctions.jar
```

3. Set the `bui.tester.engine.feature.level` property to `local` or `full`. See Configuring Project Tester Options on page 31.
4. Save the file. (Note that the next task also modifies this file.)

Task D Add the Custom Function JAR Path to the Decision Manager File Code Generation Property

Enables Decision Manager to generate and compile the java code for testing in Decision Manager.

1. Open the *BE_HOME*`/DecisionManager/configuration/bui-config.tra` file.
2. Add the fully qualified path to the custom functions JAR files in the `tibco.bui.codegen.prepend_classpath` property. For example:

```
tibco.bui.codegen.prepend_classpath=C:/custom/customfunctions.jar;
c:/custom/morefunctions.jar
```

3. Save the file.

Task E Add the Custom Function JAR Path to the RMS Property File

Enables RMS to generate and compile the java code for the production system.

1. Open the *BE_HOME*`/rms/bin/be-rms.tra` file.
2. Add the fully qualified path to the custom functions JAR to the `tibco.env.CUSTOM_EXT_PREPEND_CP` property. For example:

   ```
   tibco.env.CUSTOM_EXT_PREPEND_CP=C:/custom/customfunctions.jar
   ```

Chapter 3 Configuring User Authentication

This chapter explains how to choose and configure two authentication modules. It also explains how you can use your existing JAAS login module instead of the one provided.

Topics

User Authentication Overview

When configuring user authentication, you choose which authentication option you want to use and configure that option. You can also use a different login module if you don't want to use the module shipped with the product.

Authentication Options

RMS provides two options for user authentication.

File Based Authentication This method authenticates a user against user data stored in a file based repository. This method is not recommended for production purposes.

LDAP Authentication This method authenticates users against a directory server using LDAP as a protocol. RMS can leverage this information to authenticate users. The role information is configured through an LDAP attribute (this is directory server specific) like the `nsroledn` attribute in Sun Java System Directory Server.

Pluggable JAAS Login Module

Java Authentication and Authorization Service (JAAS) is a pluggable part of the Java security framework. User authentication is performed using a JAAS login module.

You can substitute a different implementation of the JAAS login module than the one provided. To configure the product to use your implementation, specify the location of your JAAS login configuration file using the following property in the `be-rms.tra` file:

```
java.property.java.security.auth.login.config
```

As another option, you may want to add the provided login module to your existing JAAS login configuration file (thus providing multi-stage authentication). If so, specify the location of the file in the above property.

Configuring User Authentication

This section explains how to select file-based authentication or LDAP-based authentication, and how to configure each authentication option.

Configuring File-Based Authentication

In file-based authentication, you define a list of user names, passwords, and roles in a file called (by default) `users.pwd` file. This file is commonly referred to as the password file.

Task A Configure RMS Properties

1. Open the *BE_HOME*`/rms/bin/be-rms.tra` file.
2. Set `rms.auth.type` to `file`
3. As needed specify the location and name of the password file using the property `rms.auth.file.location`.

See Table 8, Authentication Configuration Properties, on page 41 for details about these properties.

Task B Configure the Password File

1. Open the password file. As shipped, the file is located in *BE_HOME*`/rms/examples/users.pwd`.

   The name and location are configurable, as described in the notes for the `rms.auth.file.location` property. See Table 8, Authentication Configuration Properties, on page 41.

2. Add each user on a separate line using this format:

   Username`:`*password*`:`*role*`,`*role*`,`*role*`;`

   For example:

   ```
   Mark:A31405D272B94E5D12E9A52A665D3BFE:BUSINESS_USER,APPROVER;
   James:21232f297a57a5a743894a0e4a801fc3:RULE_ADMINISTRATOR;
   ```

   Do not use spaces.

It is recommended that you encrypt the password with MD5 (Message-Digest 5) hashing algorithm.

Configuring LDAP-Based Authentication

These are summary instructions only. Completing the configuration requires knowledge of the LDAP protocol.

To Configure LDAP-Based Authentication

1. Open the *BE_HOME*`/rms/bin/be-rms.tra` file.
2. Set `rms.auth.type` to `ldap`
3. Specify values for all the LDAP properties shown in Table 8 on page 41.

See Table 8, Authentication Configuration Properties, on page 41 for details about these properties.

Authentication Property Reference

Table 8 Authentication Configuration Properties

Property	Notes
`rms.auth.type`	Specifies the authentication mechanism. Allowable values are `file` or `ldap`. Default is file.
`rms.auth.file.location`	Specifies the name of the password file. By default the filename is `users.pwd`. You can specify a subdirectory in addition to a filename. Place this file (and optional subdirectory path) in the appropriate location: Single-project mode: `rms.project.location/rms.project.name/config` Multiple-project mode: `rms.projects.baselocation`. See Table 4, RMS Server Configuration Properties, on page 18 for details about the properties used to specify the above paths.
`rms.auth.ldap.host`	Specifies the domain name of the host for LDAP authentication.
`rms.auth.ldap.port`	Specifies the port for LDAP authentication.
`rms.auth.ldap.adminDN`	Specifies the base distinguished name (DN) for admin login. For example, `cn=Directory Administrators, dc=na, dc=tibco, dc=com.`
`rms.auth.ldap.adminPassword`	Specifies the password for the LDAP administrator DN.
`rms.auth.ldap.baseDN`	Specifies the base tree in LDAP under which users can be searched. For example, `dc=na`, `dc=tibco`, `dc=com`.
`rms.auth.ldap.roleAttr`	Specifies the name of the attribute used by the LDAP server for role information of a user. Default value is `nsroledn` (for Sun Java Directory Server).
`java.property.java.security.auth.login.config`	Provides the absolute location for the login module configuration used by JAAS. See Pluggable JAAS Login Module on page 38 for more details. The value as shipped is: `C:/tibco/be/3.0/rms/config/jaas-config.config`

Chapter 4 Creating an RMS Project And Starting RMS

This chapter explains how to create an RMS project and start RMS.

Topics

Creating a Rules Management Server Project

See Chapter 2, RMS and Decision Manager Configuration, on page 13 and ensure you have completed basic configuration before beginning.

Task A Create the RMS Project Directories

Under the base location directory (or the preconfigured project directory if you are using single project mode), create the following project directories (multi-project mode example shown):

Names shown below are default names. See Configuring RMS Server Properties on page 16 for configuration properties that enable you choose single-project mode and to change the default locations and names of project files. See RMS Project Directories and Files on page 14 to understand the directory structure.

```
/BaseLocation
  /RMSProjectDir
    /bin
    /config
    /data
    /decisiondata
    /deployment
    /workspace
```

Task B Create VRFs in the BusinessEvents Project and Build the EAR

A virtual rule function is a rule function with the Is Virtual checkbox checked. You provide arguments but no body (the body is provided by the decision table).

1. Create the VRF and configure the rest of the BusinessEvents project.

   This procedure is documented in *TIBCO BusinessEvents User's Guide*. See Creating Rule Functions in Chapter 9, Working With Rules and Functions.

2. Build the EAR file.

   This procedure is documented in *TIBCO BusinessEvents User's Guide*. See Chapter 26, Deploytime Configuration.

3. Copy the EAR file to in the *RMSProjectDir*`/bin` directory.

Task C Create the Access Control File

Every project requires its own access control list. The list can be configured to set permissions on the ontology entities and rules in the BusinessEvents project. The permissions are used in Decision Manager.

1. Configure the access control file as explained in Chapter 3, Configuring User Authentication, on page 37.
2. Name the access control file *ProjectName*`.ac`
3. Place the file in the *RMSProjectDir*`/config` directory.

Task D Start RMS

Starting the RMS server makes the RMS project available to Decision Manager users. Users can check out the project and create their own decision project.

However you can do two additional tasks as shown next

See Starting the RMS Server on page 46.

Task E Create a Domain Model File and Test Data File (As Desired)

You can create a domain model that defines allowable values for some or all of the properties in the project. You can also create sets of sample data for use in testing.

To provide these additional resources, you check out the project using Decision Manager and use the user interface to create the files. Then you copy them from the decision project location to the RMS project location. Project locations are configurable. See Configuring RMS Server Properties on page 16.

1. Start Decision Manager and create a decision project from the RMs project (see Creating a Decision Project on page 58).
2. To create a domain model file do the following.
   a. Define the domain model as explained in Defining Domain Models on page 67.
   b. Copy the domain model file from the *DecisionProject*`/config` directory to the *RMSProject*`/config` directory (or whatever directory you defined in the `be-rms.tra` file). Name the file *RMSproject*`.dm`.
3. To create test data files, do the following.
   a. Define the test data as explained in Testing Decision Tables on page 81.
   b. Copy the test data files from the *DecisionProject*`/data` directory to the *RMSProject*`/data` directory (or whatever directory you defined in the `be-rms.tra` file.)

Starting the RMS Server

Starting the RMS server makes RMS projects available to Decision Manager users. It also enables approval workflow features.

To start RMS, do one of the following:

- Execute *BE_HOME*`/rms/bin/be-rms.exe` (`or be-rms.sh`, depending on your operating system).
- (Windows only) Select Start > All Programs > TIBCO > TIBCO Business Events Enterprise Edition *version* > Start Rules Management Server.

When using the Berkeley database as a persistence store for RMS, if an error message such as *Payload Type Not registered* is seen (this appears only in exceptional circumstances), clean up the Berkeley database and restart RMS.

Chapter 5 Configuring Access Control for a Project

This chapter describes how the access control system works, the types of resource specifications that are permitted, and how to grant permissions to different users.

Access control is set on each RMS project.

Topics

Configuring Access Control—Overview

For each RMS project, you set up an access control file. In the access control file, you group the decision project resources as desired, giving each group (or individual resource) an ID, and then you assign permissions to each user role, using those IDs.

See also Configuring Workflow Status Permissions for User Roles on page 25. This configuration is done at the server level, not at the project level.

Establishing the User Roles

Access is defined using roles. If file-based authentication is used, roles are defined and assigned to users in the password file (by default called `users.pwd`). If LDAP-based authentication is used, roles are defined and assigned to users in the LDAP directory.

You add one entry element for each user role. See Structure of the Access Control File on page 50 for details.

See User Authentication Overview on page 38 for details.

You must use only the roles defined in the password file or LDAP directory (depending on authentication type used) when configuring the access control file.

Guidelines for Configuring Access Control

You can use two general approaches to setting permissions. The general aim is to simplify the setup, minimizing the number of permissions you have to set in the access control file.

Allow everything and specify exceptions

One approach is to grant wide permissions using large resource groupings, and then selectively deny permissions within those groupings.

For example, suppose you define two resources as follows:

```
<resource name="/Concepts/*" id="AllP" type="PROPERTY"/>
<resource name="/Concepts/Person/CustID" id="CID"
type="PROPERTY"/>
```

The first `resource` element defines a resource group consisting of all concept properties in the /Concepts project folder. The second element specifies one property in one concept. (The setup details are explained later in the chapter.)

Then you define permissions for using those resources, for a role named `CallCenter`:

```
<permission resourceref="#AllP">
  <action type="create">ALLOW</action>
 </permission>
 <permission resourceref="#CID">
  <action type="create">DENY</action>
 </permission>
```

With these settings, you give users with the `CallCenter` role the `create` permission for all properties in the `/Concepts` directory except the `custID` property.

An example of an access control file giving full permissions is provided in the credit card application example, located in the following directory:

BE_HOME`\rms\examples\CreditCardApplicationthe CreditCardApplication`

Deny everything and specify exceptions

The other approach is to deny all permissions (which is the default setting for all permissions) and then give permissions to specific resources or groups of resources as needed.

Mixing these approaches

You can mix these two approaches in one access control file. For example, you can give broad permissions to one project folder, and then specify exceptions within that folder. For another folder you might give permissions selectively.

Structure of the Access Control File

The access control file is an XML file with the following elements:

```
<acl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ACL.xsd">
<resources>
  <resource id="id" type="ResourceType"/>
  <resource id="id" name="ProjectPath" type="ResourceType"/>
  . . .
</resources>

<entries>
  <entry>
    <role name="RoleName"/>
    <permissions>
      <permission resourceref="#id">
        <action type="ActionType">[ALLOW|DENY]</action>
      </permission>
      . . .
    </permissions>
  </entry>
  . . .
</entries>
</acl>
```

The `entries` element contains one `entry` for each role. For each role, you define one set of `permissions`. Each `permission` has of the following attributes

- The **`resourceref`** attribute references a **resource ID** defined in the resources element. It identifies a resource or set of resources.
- The **`name`** attribute specifies the **project path** to the resource or resources. (The `name` attribute is not used when you specify permissions for an entire resource type.)
- The resource **`type`** attribute specifies what types resources in the specified project path are included in the permission.
- The action **`type`** attribute specifies an **action type**, for example, `create`.

See Table 9, Resource Types and Action Types, on page 54.

Permissions—ALLOW and DENY

The value of the `action` element is one of the key words `ALLOW` or `DENY`. It determines whether the specified permission is given denied.

`DENY` is the default value. You only need to set the `DENY` value explicitly when you have given `ALLOW` permissions at a higher level, and want to make individual exceptions within that broad scope.

The values `ALLOW` and `DENY` are case sensitive. Use all capitals.

Working with Access Control Files

This section explains the elements used to define access control. After reading this section, you will be able to set up your access control file. Examples shipped with the product contain access control files you can use as models.

Creating and Modifying an Access Control File

You can create or modify an *RMSProject*`.ac` file using any XML editor.

If your user role has permission to do so, you can also create or modify an access control file in Decision Manager.

To create an access control file in Decision Manager, select **Access** > **New**.

To modify an existing access control file, select **Access** > **Open**.

Location of Access Control Files

The access control file for an RMS project must be placed in the project's `config` folder and it must be named using the format *ProjectName*`.ac`.

See RMS Project Directories and Files on page 14. also see and Choosing Single Project or Multiple Project Mode on page 16 to understand where the project folder or folders are located.

Specifying and Grouping Decision Project Resources

In the `resources` element, you group the project resources in whatever way supports the permissions you want to set and give each grouping or individual resource an ID. You use the ID when setting the permissions.

Using Resource Type as a Filter

How you specify the resource group is partly determined by the resource type attribute. The resource `type` can act as a filter. For example, suppose in the `name` attribute you specify a folder that includes events and concepts. If you set the `type` attribute to `"CONCEPT"` then the ID associated with this grouping is used to set permissions only on the concepts in that folder (and its subfolders).

You could create a second grouping whose `type` specifies `"EVENT"` so that you can set permissions on events in that folder branch separately.

Specifying an Individual Resource

To specify an individual resource, provide the *project path* to the resource in the `name` attribute. The project path is the folder path to the ontology entity, as seen in the Explorer panel. The example below shows how to specify an ID that is associated with the `FirstName` property of the `Person` concept:

```
<resource name="/Concepts/Person/FirstName" id="FN"
type="PROPERTY"/>
```

Grouping Resources Using Wildcards

You can associate groups of resources with an ID using the wildcard character in the project path. The asterisk (*) is used as the wildcard character. For example:

```
<resource name="/someFolder/* "id="AllP" type="PROPERTY"/>
```

Grouping Resources by Resource Type

The broadest resource grouping is provided by setting permissions at the level of resource type. This method groups all resources of that type in the project. To set a resource type resource group, you associate an ID with a resource type, and you do not use the `name` attribute:

`<resource id="`*ID*`" type="`*ResourceType*`"/>`

For example: `<resource id="C" type="CONCEPT"/>`

See Table 9, Resource Types and Action Types, on page 54 for a list of resource types, and the action types that are valid for each resource type.

Defining Permissions

By default, all permissions are denied. If a certain permission is not explicitly given to a role, then the role does not have the permission. This approach ensures unauthorized users do not accidentally gain access to restricted resources.

Permissions are not hierarchical. That is, a `create` permission does not imply a `modify` permission or a `delete` permission. All privileges are mutually exclusive.

For each resource type there is a predefined set of action types (see Table 9, Resource Types and Action Types, on page 54). You must grant permission separately for each action type. For example, you would add four permission elements to give a user role permissions to `create`, `read`, `modify`, and `delete` a specified group of resources of a certain type.

In the Decision Manager application, you cannot create, modify, or delete properties, rulesets, concepts, and rule functions.

Resource Types and Corresponding Action Types

For each type of resource, you define permissions for the action types available for that type of resource, as shown in Table 9:

Table 9   Resource Types and Action Types

Resource Type	Allowable Action Types	Notes
`CONCEPT`	read	
`EVENT`	read	
`RULESET`	read	
`RULE`	`read`	This permission enables users to view the source for a rule.
`RULEFUNCTION`	read, `add_impl`, `del_impl`, invoke	`add_impl` enables a user to add decision table implementations. `del_impl` enables a user to delete decision table implementations for VRFs.
`PROPERTY`	read	The permission applies to the properties of the specified resource or resources.
`DOMAINMODEL`	create, read, modify, delete	The permission applies to domain models in the specified resource or resources.
`DECISIONTABLE`	read	The permission restricts what decision tables a user can view in the project explorer. (Other permissions for decision tables are set on the resources used in the decision table.)
`PROJECT`	checkout, update, commit	This resource type applies to actions taken at the project level (and not to all resources within a project).
`CATALOGFUNCTION`	invoke	Users see only the functions for which their role has invoke permission. Validation errors are thrown if other functions are written manually.
`FOLDER`	read	Controls access to named folders.

Chapter 6 Working With Decision Manager Projects

This chapter deals with tasks at the decision project level. For activities relating to working with individual decision tables see Chapter 7, Working with Decision Tables, on page 63.

Topics

Working with Decision Projects—Overview

Decision Manager can connect to RMS on any machine accessible to the machine where Decision Manager is running and you can check out a decision project and work with it.

Depending on your roles, and permissions defined in the RMS project access control file, you may not have permission to do all of the documented tasks.

Working with decision projects includes the following activities:

- Creating, opening, refreshing (updating), and saving a decision project. Committing a project and checking on its status. These project-level activities are explained in the current chapter.
- Creating and building decision tables for the project. Importing, exporting and deleting decision tables. See Chapter 7, Working with Decision Tables, on page 63.
- Creating a domain model for decision table properties. See Defining Domain Models on page 67
- Creating test data for decision tables and testing them. See Testing Decision Tables on page 81

When you delete decision tables and commit the project for approval, those tables are permanently deleted from the RMS project when the project changes are approved. They are no longer available for checkout by any decision project.

Navigation Tips

You can open different views (windows that show information) and editors (windows where you can take actions of various kinds).

- **Window > Show View** allows you to reopen some of the panels (Project Explorer, Argument Explorer, Overview, and so on) if you have closed them.
- **Window > Navigation** allows you to navigate through various editors or panels.

Starting and Logging in to Decision Manager

See Chapter 3, Configuring User Authentication, on page 37 for configuration details.

To log in to example projects as the rule administrator, enter user name `admin` and password `admin`. To log on as a business user, enter user name `jdoe` and password `jdoe`.

To Start Decision Manager and Log in

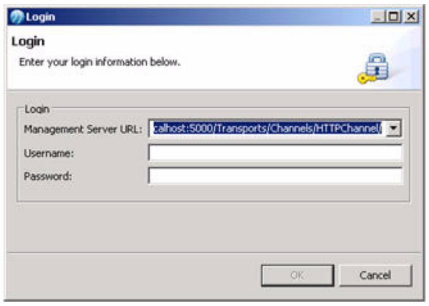
1. Start Decision Manager in one of the following ways:

   — *BE_HOME*`/DecisionManager/DecisionManager.exe` (or `DecisionManager.sh`, depending on your operating system).

   — Select Start > All Programs > TIBCO > TIBCO Business Events Enterprise Edition *version* > Decision Manager.

   The Splash screen and Welcome screen display. The Welcome screen contains links to BusinessEvents documentation.

2. Close the Welcome screen. The application workspace displays.
3. From the File menu, select **Login**. You see the The Login dialog:



Login

Enter your login information below.

Login

Management Server URL: calhost:5000/Transports/Channels/HTTPChannel/

Username:

Password:

4. To select a specific RMS server, enter its URL or select the correct URL from the drop-down list.

RMS servers can be installed on any machine accessible to Decision Manager. If multiple RMS servers are in use, you can connect to the one you want to work with using its URL. Ask an administrator for the URL if you don't know it and it is not in the list already

5. Enter a valid username and password and click **OK**. You see a confirmation message. Click **OK**. The Decision Manager menu options enabled for your user role display.

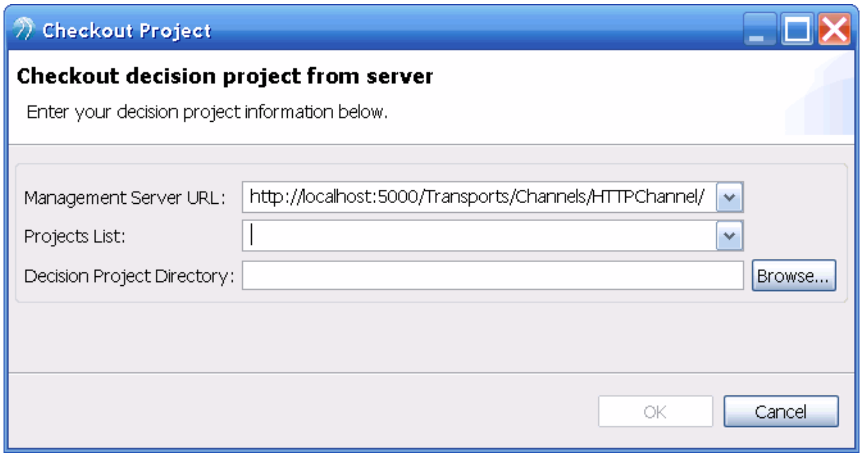
Creating a Decision Project

To create a decision project, you select an RMS project to check out and specify a location for the local decision project that is based on it. Then you can work with its decision tables. The server sends the files needed to work with the project to the directory you specify.

`Creating a Decision Project Using a Utility` The `dpgen.bat` batch file enables you to generate a decision project without checking it out from RMS. You can work on this locally-generated project in Decision Manager and commit it when you are ready. This batch file is located in the *BE_HOME*`/rms/bin` directory.

To Create a Decision Project

1. From the Project menu, select **Checkout**. You see the Checkout Project dialog:



Checkout decision project from server

Enter your decision project information below.

Management Server URL: http://localhost:5000/Transports/Channels/HTTPChannel/

Projects List:

Decision Project Directory:

2. Select the RMS Server URL from the list or add another RMS URL to the list if you need to check out projects from a different RMS server.
3. In the Projects List field, select the project you want to work with from the drop-down list. (See Chapter 4, Creating an RMS Project And Starting RMS, on page 43 for details about creating projects that appear in this list.)
4. In the Decision Project Directory field, browse to and select a directory on the local file system.
5. Click **OK**. You see a confirmation message. Click **OK** again. The decision project is created in the specified directory.

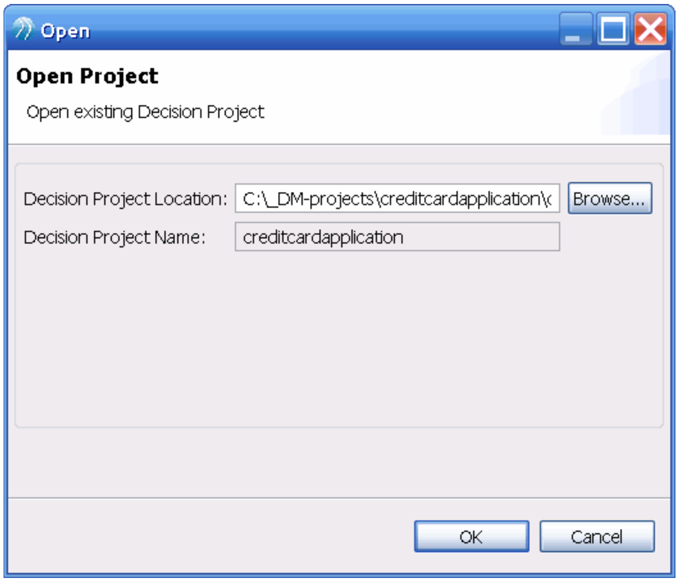
The Project Explorer panel displays the project ontology (concepts, events, rules and so on, according to your role's permission to view those resources).

Opening an Existing Decision Project

If you have saved a Decision Manager project locally you can open it as follows. Note that there is no need to log in to open a local decision project.

To Open a Decision Project

1. Start Decision Manager and select **File > Open**. You see the Open Project dialog.
2. Browse to the project folder for the decision project you want to open, and select *projectname*`.dp` file.



Open

Open Project

Open existing Decision Project

Decision Project Location: C:_DM-projects\creditcardapplication\c

Decision Project Name: creditcardapplication

3. Click **OK**.

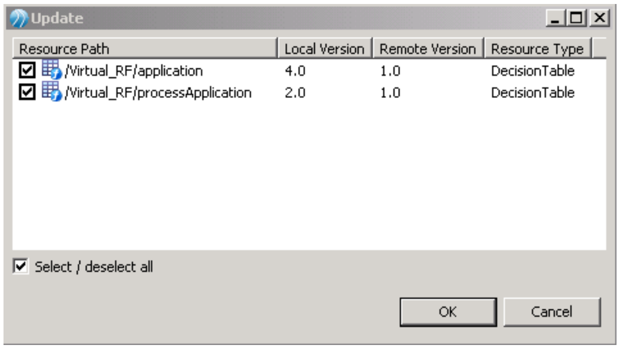
Updating (Refreshing) a Decision Project

You can update the decision projects on your local machine with the most recent changes from the corresponding RMS projects.

To Update a Decision Project

1. Check out the project as described in , Creating a Decision Project, on page 58 and work From the Project menu, select **Update**.

   The Update dialog lists decision tables that were updated in the RMS project after you checked out the project.



Resource Path	Local Version	Remote Version	Resource Type
☑ /Virtual_RF/application	4.0	1.0	DecisionTable
☑ /Virtual_RF/processApplication	2.0	1.0	DecisionTable

2. Select the decision tables you want to copy to your local decision project. Selected decision tables overwrite your local copies.
3. Click **OK**

The following local decision project components are also updated if the RMS server components changed since you checked out the project:

- The access control file (not in human readable form).
- The domain model file.
- Decision tables that were added to the RMS project after you checked out the project.

A Progress Information dialog is displayed while the tables are updated.

Committing Decision Tables for Approval

When you are satisfied with your decision table or tables, you then commit them for approval and deployment to the BusinessEvents production system.

When you commit decision tables, a copy of the submitted tables is saved in the *RMSProject*`/workspace` directory, along with the name of the user who submitted it, the version number, and a status = `Approval Pending`.

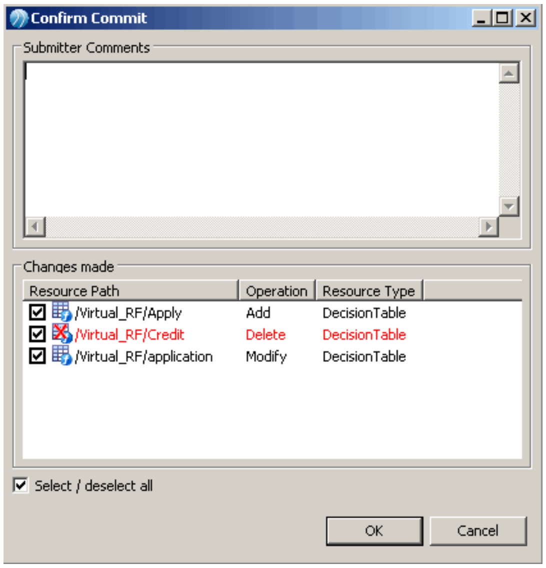
A Request ID is assigned to each commit request. This ID is used in other dialogs to allow you to view details on the request. See Checking the Status of a Committed Project on page 62.

A task is created for the Approver role to view the check-in requests that await approval (see Chapter 8, Working with the Approval Process, on page 93).

When you commit a decision table for approval, any domain model changes you have made (if any) are also committed. Upon approval, the updated domain model replaces the master copy in RMS.

To Commit Decision Project Resources for Approval

1. Select **Project > Commit**. The Confirm Commit dialog lists the newly added, modified, and deleted (red) resources.



	Resource Path	Operation	Resource Type
✓	/Virtual_RF/Apply	Add	DecisionTable
✓	/Virtual_RF/Credit	Delete	DecisionTable
✓	/Virtual_RF/application	Modify	DecisionTable

2. In the Submitter Comments panel, type any comments you want the Approver to read. Then select the decision tables you want to commit, and click **OK**.

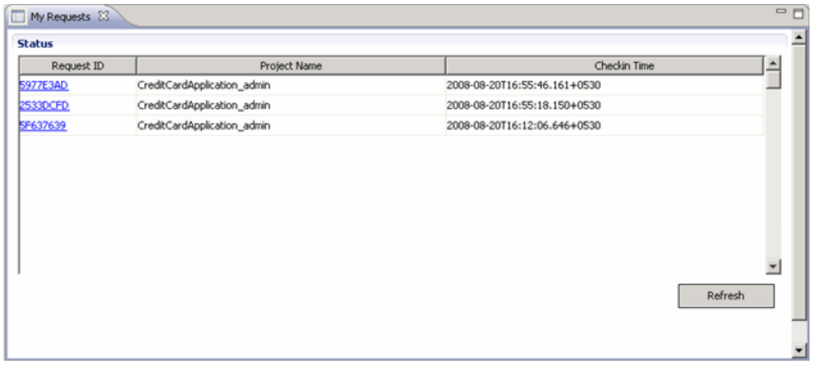
Checking the Status of a Committed Project

You can check the status of a project you committed for approval. You can see whether your project was approved or rejected, or if it was reviewed at all, and you can view comments from the Approver if any.

It is your responsibility to take appropriate action depending on the approval status.

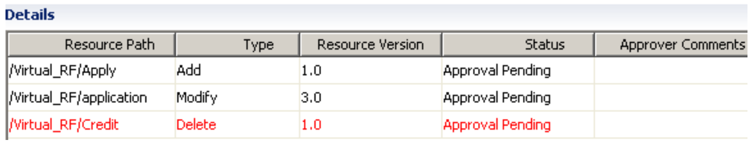
To Check Project Status

1. From the Project menu, select **Show Status**. The My Requests editor displays the list of check-in requests you have submitted.



Request ID	Project Name	Checkin Time
5977E3AD	CreditCardApplication_admin	2008-08-20T16:55:46.161+0530
2533DCFD	CreditCardApplication_admin	2008-08-20T16:55:18.150+0530
5F637639	CreditCardApplication_admin	2008-08-20T16:12:06.646+0530

2. In the Request ID column, click a **Request ID**. The Details panel in the lower part of the editor displays details about that check-in request. For example:



Resource Path	Type	Resource Version	Status	Approver Comments
/Virtual_RF/Apply	Add	1.0	Approval Pending	
/Virtual_RF/application	Modify	3.0	Approval Pending	
/Virtual_RF/Credit	Delete	1.0	Approval Pending	

Deletion requests are shown in red.

3. Click **Refresh** to ensure the list is current.

Chapter 7 Working with Decision Tables

This chapter deals with working with decision tables in a decision project. For tasks done at the decision project level, see Chapter 6, Working With Decision Manager Projects, on page 55.

Topics

Creating, Opening, and Saving Decision Tables

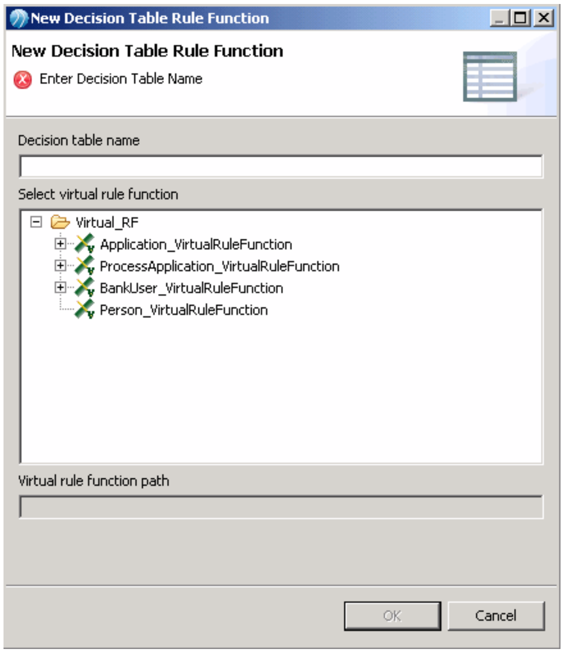
Give each table a unique name. Do not use the same name for different tables in the same project, even if they implement different VRFs.

You can create a decision table from the file menu, or by selecting a VRF in the Project Explorer panel. You can open a decision table that already exists in the project, as explained in the following procedures.

You can identify a VRF (virtual rule function) in the project explorer view by its icon. A small letter "v" for "virtual" is added to the standard rule function icon:

To Create a Decision Table From the File Menu

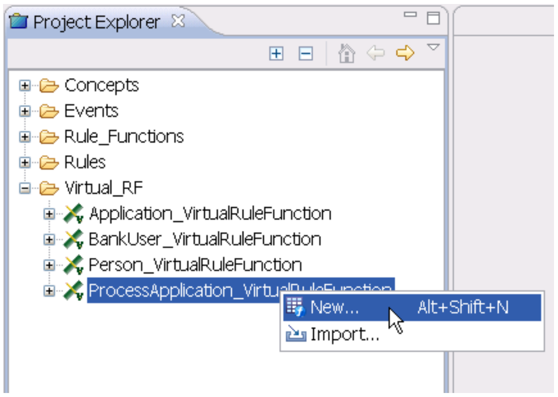
1. Create or open a decision project. See Creating a Decision Project on page 58 or Opening an Existing Decision Project on page 59 for details.
2. From File menu Select **New > Decision Table**. New Decision Table Rule Function dialog displays.


New Decision Table Rule Function
New Decision Table Rule Function
Enter Decision Table Name
Decision table name
Select virtual rule function
Virtual_RF
Application_VirtualRuleFunction
ProcessApplication_VirtualRuleFunction
BankUser_VirtualRuleFunction
Person_VirtualRuleFunction
Virtual rule function path
OK
Cancel

3. In the Decision table name field, enter a name for the new decision table.
4. In the Select virtual rule function panel, expand the list and select the VRF for which to create the decision table. Then click **OK**.

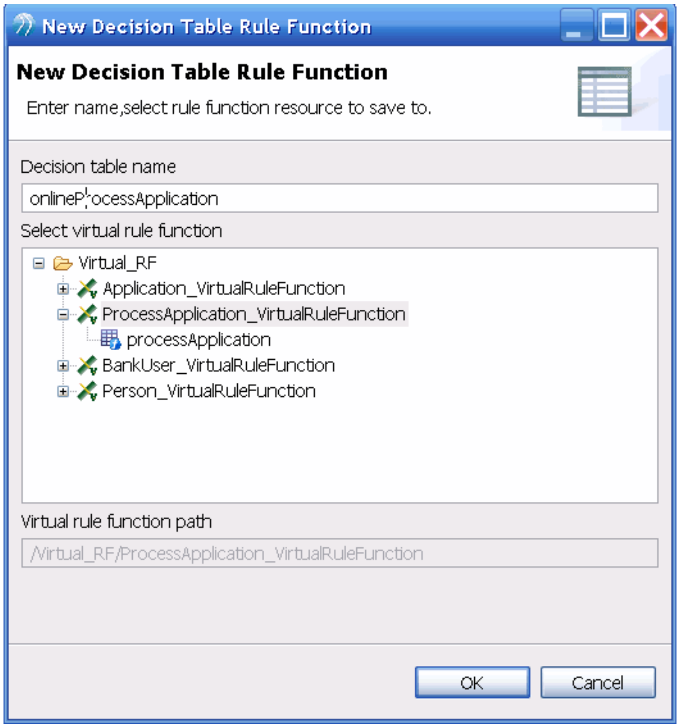
To Create a Decision Table from the Project Explorer

1. In the Project Explorer panel, select a virtual rule function (VRF) for which to create a new Decision Table.
2. Right-click the selected VRF and select **New**.


Project Explorer
Concepts
Events
Rule_Functions
Rules
Virtual_RF
Application_VirtualRuleFunction
BankUser_VirtualRuleFunction
Person_VirtualRuleFunction
ProcessApplication_VirtualRuleFunction
New... Alt+Shift+N
Import...

The New Decision Table Rule Function dialog displays, showing the selected VRF.

3. In the Decision table name field, enter a name for the new decision table and click **OK**.



New Decision Table Rule Function

New Decision Table Rule Function

Enter name,select rule function resource to save to.

Decision table name

onlineProcessApplication

Select virtual rule function

```
⊟ Virtual_RF
    ⊞ Application_VirtualRuleFunction
    ⊟ ProcessApplication_VirtualRuleFunction
        processApplication
    ⊞ BankUser_VirtualRuleFunction
    ⊞ Person_VirtualRuleFunction
```

Virtual rule function path

/Virtual_RF/ProcessApplication_VirtualRuleFunction

To Open an Existing Decision Table

1. Open a decision project.
2. In the Project Explorer panel, expand the virtual rule function (VRF) to see decision tables created for that VRF. Double-click a table name to open it.

To Save a Decision Table

Do any of the following as needed:

- Select **File > Save**.
- Select **File > Save As** and enter a different name to save a copy of the decision table with a different name.
- Select **File > Save All** to save all open decision tables.

Defining Domain Models

A domain model restricts the values users can enter in decision table cells. For example, instead of typing text for a certain concept property, a user may have to pick a value from a list or work within a predefined range.

When you create the domain model for a property, you define the acceptable values. You can create domain models for concept properties.

You can't create domain models for properties of contained concepts or referenced concepts. (Containment and reference are relationships between concepts, defined in the BusinessEvents project. See *TIBCO BusinessEvents User's Guide* for more details.)

Who Defines Domain Models

Managing domain models is often an administrative task not given to business users, but different enterprises have different needs. Use access control to determine who can perform this task.

Administrators can create the domain model when setting up an RMS project for use by business users who check out a decision project from that RMS project. In this scenario, business users can also have permission to create or modify domain models, or they can be prevented from changing the domain model, depending on the situation.

Business users with permission can create the domain model after checking out an RMS project. It could be the business users who have primary responsibility for creating domain models, because they may be the ones who know what values to allow.

- Use access control to determine whether users can enter values freely or are restricted to using the domain model values. See Chapter 5, Configuring Access Control for a Project, on page 47.
- When you commit a decision table for approval, any domain model changes you have made (if any) are also committed.

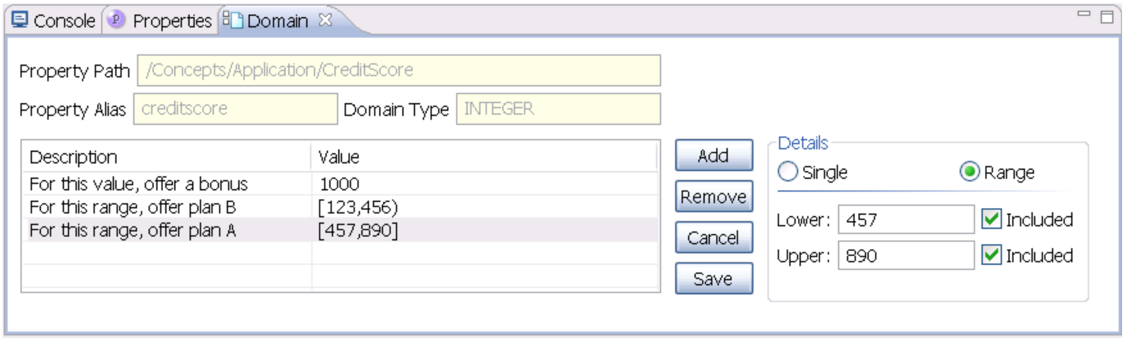
Domain Models for Primitive Data Types

The following kinds of domain models can be created for each data type. All domain model values can have optional descriptions that appear only in the domain model editor:

String A list of text values.

Boolean Two values named True or False.

Numeric (`integer`, `long`, `real`) A single value or a range, which can be inclusive or exclusive. (Note that BusinessEvents rule language uses the term `double` instead of `real`):

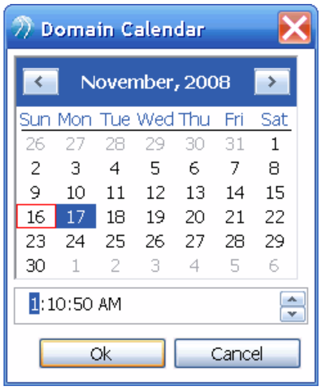


Property Path: /Concepts/Application/CreditScore
Property Alias: creditscore
Domain Type: INTEGER

Description	Value
For this value, offer a bonus	1000
For this range, offer plan B	[123,456)
For this range, offer plan A	[457,890]

Details
○ Single [✓ Range]
Lower: 457 [✓ Included]
Upper: 890 [✓ Included]

DateTime A single date and time value or a date and time range, which can be inclusive or exclusive. A date and time picker enable you to select the date and time.



Creating and Managing Master Domain Model Files

The domain model file in the RMS project is known as the master domain model. To create a domain model file for an RMS project do the following.

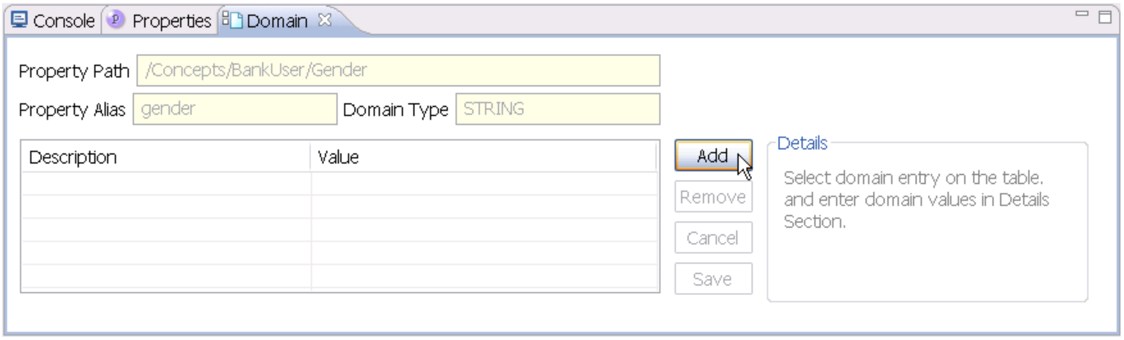
1. Check out the project as a decision project, define the domain model, and save the project. See Defining Domain Models on page 67.
2. Copy the decision project domain model file (*DecisionProject*`.dm`) file from the decision project root directory tree to the *RMSProject*`/config` directory.

The master domain model is provided with the decision project when a user checks out a project. Users with permission can then define local domain models as desired.

The directory structure is configurable. See RMS Project Directories and Files on page 14 and Configuring RMS Server Properties on page 16 for details.

Defining Domain Models

1. Check out the project for which you want to create the domain model. See Creating a Decision Project on page 58 or Opening an Existing Decision Project on page 59.
2. Select a concept property in the project explorer or argument explorer, and select the Domain tab in the properties area. (Entities appear in the argument explorer when you select a virtual rule function in the project explorer panel.)
3. Click **Add**.



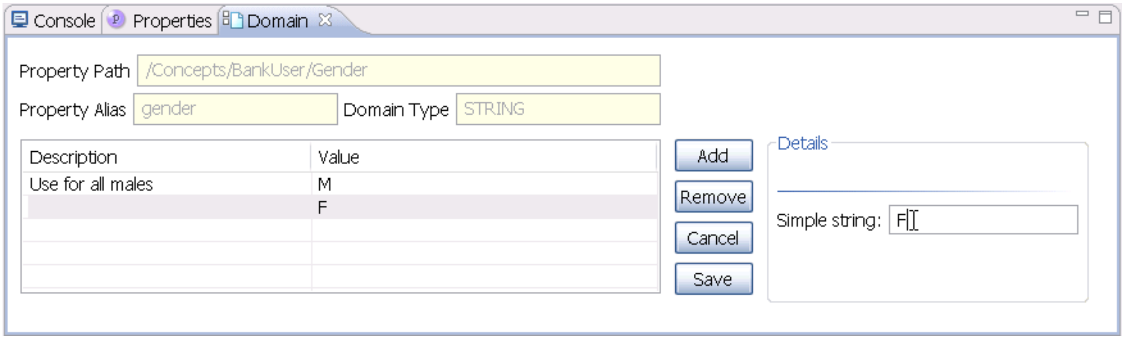
Property Path /Concepts/BankUser/Gender
Property Alias gender
Domain Type STRING

Description	Value

Select domain entry on the table. and enter domain values in Details Section.

Depending on the property's Domain Type, you can create different kinds of domain models for the property. This procedure uses a `String` property. See Domain Models for Primitive Data Types on page 67 for information about other data types.

4. In the Details panel, enter the value and click **Add**.



Property Path: /Concepts/BankUser/Gender

Property Alias: gender

Domain Type: STRING

Description	Value
Use for all males	M
	F

Simple string: F

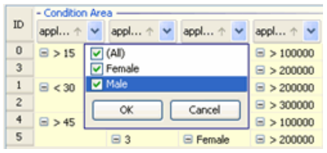
5. As desired, enter an optional description in the Description column. The description does not appear in the decision table itself.
6. Add, modify, and remove rows until you are satisfied with the domain model.
7. Click **Save**.

Using Domain Models

When entering values for a cell that has a domain model, you see a drop-down list. Select a value from the list.

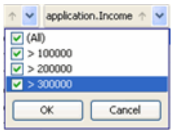
When you select multiple values, an implicit OR is applied, that is, any one of the selected values is accepted.

Below is an example of a domain for a String property:



ID	appl...	appl...	appl...	appl...
0	> 15			> 100000
3				> 200000
1	< 30			> 200000
2				> 300000
4	> 45			> 100000
5		3	Female	> 200000

Below is an example of a domain for a numeric property, allowing you to specify ranges:



Building Decision Tables

After you create or open the decision table (See Creating, Opening, and Saving Decision Tables on page 64), you can build the table or edit what was built in a previous session. This section explains the basic procedure.

A decision table uses arguments defined in the VRF, which is a virtual rule function created in the BusinessEvents project. The arguments are concepts or events or both. To build the table you drag arguments to the conditions area or the actions area as appropriate.

Each row in a decision table can be thought of as an "if-then" sentence: "If X is the case, then do Y." The "if" part of the sentence contains the condition or conditions. The "then" part of the sentence contains the action or actions.

To Build a Decision Table

1. In the Argument Explorer panel, expand the entity or entities to see their properties. (An entity is a concept or an event).
2. Drag properties to be used as conditions to the Condition Area.

   For example, suppose the table will determine whether to accept a credit card application based on a bank users's age and credit score. You would drag and drop the `BankUser` concept's `age` and `creditScore` properties from the Argument Explorer view to the Condition Area of the decision table.
3. Drag properties to be used as actions to the Action area.
4. Click the plus sign to add a row to the table.
5. Drag concept properties to table cells and define formulae as needed (using functions or allowable symbols) to form the conditions and actions for a business rule. For example, each row could define actions to take depending on the age and credit rating of a credit card applicant.

String properties cannot start with an integer unless you wrap the whole value in quotes. For example: `"12hello"` is valid.

6. When you are finished creating the decision table, click **File > Save**.

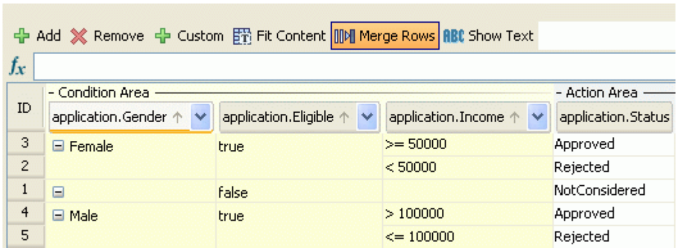
- Rows with the same conditions are merged. Actions are also merged.
- Blank cells, cells that have an asterisk (*) in them, and disabled cells are all ignored and automatically treated as true

Tips on Working With Decision Tables

This section summarizes various actions you can take to work with decision tables:

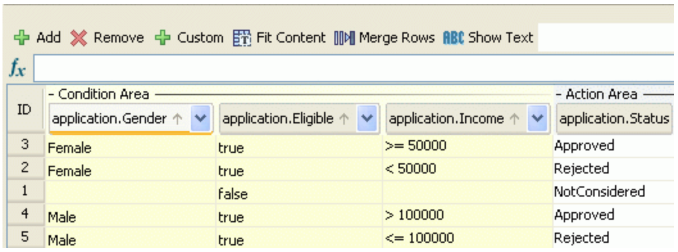
- Click **Add** to add rows. Enter the data in the new cells or drag and drop the properties from Argument Explorer.
- Click **Fit Content** to resize the columns so they fit around the text.
- You can search for certain values by entering text in the **Search** field.
- Click the **Show Text** button to see the contents of the cell, instead of just the values you are comparing.
- Click **Merge Rows** to merge two separate rows with the same data. You can toggle between merged and separated rows by clicking on this tab. If similar conditions are present in a decision table, they get merged into one cell for easier reading as with pivot tables.

  The following figure shows two rows with the same condition (same gender) merged into one row:



ID	Condition Area			Action Area
	application.Gender	application.Eligible	application.Income	application.Status
3	⊟ Female	true	>= 50000	Approved
2			< 50000	Rejected
1	⊟	false		NotConsidered
4	⊟ Male	true	> 100000	Approved
5			<= 100000	Rejected

  The following figure shows two rows with the same condition (same gender) un-merged (separated out) into two separate rows:



ID	Condition Area			Action Area
	application.Gender	application.Eligible	application.Income	application.Status
3	Female	true	>= 50000	Approved
2	Female	true	< 50000	Rejected
1		false		NotConsidered
4	Male	true	> 100000	Approved
5	Male	true	<= 100000	Rejected

- Drag and drop the catalog functions you see collapsed by default on the right side of the application. Drag and drop them to the function area (marked by the *fx*) or to the cell you are editing. For example, if you want to compare some attribute that is of type integer against the rounded value of `39.99`, you could specify a condition `< Math.round(39.99)` by dragging and dropping the Math round function from the Standard Functions panel and then entering the arguments of that function.

- To take a few actions relating to the column you have clicked on, right-click on the condition area on the properties you dragged there. For example, you can move the column, remove it, or change other field settings.
- Click on the drop-down menu on the properties dragged to either the condition or the action area to filter out which rows to show based on the values. For example, if you want to see just the Account.AccountType where AccountType is "current" and you want to filter out all the other rows that do not have this value, selecting the "current" value from the drop-down list shows you all the values you have entered so far in that column.
- Click **Remove** to delete rows. If you add more rows after this operation, the rows IDs are not reused.
- Right-click on column headers and choose **Remove** to delete any conditions or actions from the table. You cannot remove the last condition column because a minimum of one condition column must exist if you have any action columns. When you remove columns, certain rows may also get merged if they now share the same values in their condition cells. If this happens, if you have non-custom actions and the rows have the same priority, the last row to be merged has its actions merged and the others deleted.
- Select **Table > Show Property**, to see meta-data of the entire decision table. You can only modify the effective date and the expiration date as described in Setting Effective Dates on page 73.

Setting Row Priorities

The order in which rows are evaluated in a decision table is not specified. If you want to control which actions are taken first, you can change those rows to have a higher priority. Click anywhere inside that row. In the Rule Properties View, you can change the priority. All actions with higher priorities are executed before actions with lower priorities. Ten is the lowest priority and one is the highest.

Setting Effective Dates

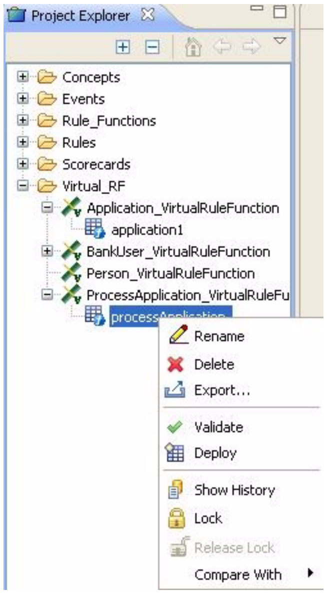
To make a decision table execute only between certain dates, you can set effective and termination dates to it. For example, if you have certain rules that should only take effect on a certain date, you can set the effective date and no termination date. Select the **Calendar** wizard available in the Property View to specify a date and time.

These rules apply:

- If no effective or expiration dates are specified, the decision table is executed every time it is invoked.
- If only an effective date is specified, the decision table is executed only if the current time is on or after the effective date.
- If both effective and termination dates are specified, the decision table is executed only if the current time is between these dates (inclusive).
- It is not valid to have only an expiration date and not an effective date. Thus, if no effective date is specified, the expiration date is ignored. This means the table is executed every time it is invoked.

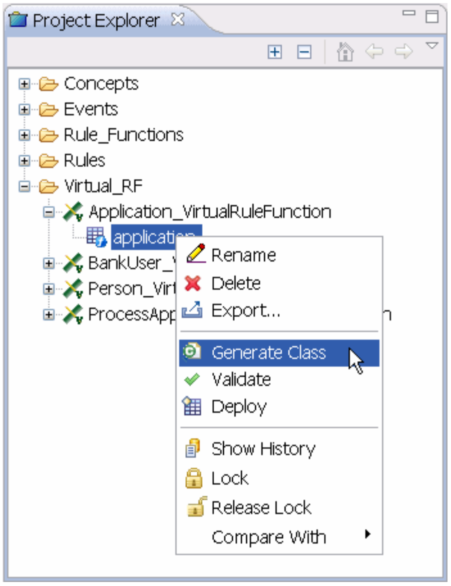
Context Menu Options for Decision Tables

You can perform certain actions on decision tables in the Project Explorer view using a context menu. Right click on one or more decision tables to see this menu. It enables you to rename, delete, export, generate class files, validate, deploy, show history, lock, release lock, and compare decision tables without having to open the tables. You can compare two different decision tables existing on the local machine or can compare a decision table with its latest copy on RMS.



The Show History, Lock, Release Lock, and Compare With (Approved Copy) options are enabled only when the user is logged on to Decision Manager.

Generate Class Option



A Generate Class option appears when the `bui.gen.class.option` property is set to `true` in `bui-config.tra`. Use this option to generate the class file for the selected decision table. Use the class, for example for testing purposes. For configuration instructions, see Configuring Decision Manager Properties on page 21.

Understanding Column Types

There are four different types of columns in decision tables. In a decision table, all the cells in a column must be of the same type. The types of columns are:

- non-Custom Condition
- Custom Condition
- non-Custom Action
- Custom Action

All cells that are not Custom apply to a single decision table argument (that is, a property that is dragged from the Argument Explorer) that is associated with their column.

Custom Conditions and Actions

You can add conditions that are not specific to any of the properties you have dragged to the condition area, and you can take actions not specific to the properties you have dragged to the action area. To do this use custom conditions or actions. Click Custom and choose **Add Custom Condition** or **Add Custom Action**. In the newly added column, enter any valid TIBCO BusinessEvents statement. The header area for each custom column is blank.

non-Custom Condition Columns

There are two sub-types for non-Custom Condition cell types:

Domain Model Condition

Check one or more boxes in the domain model drop-down. If you edit the text of the cell, the cell is no longer considered a Domain Model Condition and the domain model checkboxes are unchecked the next time you click on the cell.

The access security system enforces whether you are allowed to create non-Custom functions that do not use the domain model made available to you via the drop-down menus you see when you click on a table cell.

non-Domain Model Condition

To fill a a decision table cell of this type, enter an optional operator (= is assumed if not given) followed by any valid BusinessEvents rules language expression that can be compared with this column's argument using the operator provided.

The one difference from BusinessEvents rules language for non-Custom Conditions is that if you type an identifier (that is, a variable name, function name, and so on) that cannot be resolved as a local variable or argument name, it is treated as a String.

The operator itself is not considered part of the BusinessEvents rules language expression. The operator can be a decision table-only operator (= for equality and <> for inequality) or a BusinessEvents operator.

Custom Condition Columns

A BusinessEvents rules language expression that evaluates to a boolean value.

This is one line in the Designer Rule Editor condition area without the final ';'. Thus, you can just type "Math.round(x)". See Custom Conditions and Actions on page 76 for more information about custom conditions.

non-Custom Action Columns

A BusinessEvents rules language expression that can be assigned to this column's decision table argument. The result of the expression in this cell is always assigned to this column's decision table argument. Assignment to other properties is not allowed in this type of cell.

Custom Action Columns

This is an action area in a cell for the entire Rule Editor. In this cell you cannot make any assignments. However, you can call a function that does the assignment. If you want to make an assignment, you can do it explicitly. See Custom Conditions and Actions on page 76 for more information about custom actions.

Locking and Unlocking Decision Tables

While working on a decision project you can lock a decision table. Lock and release lock functionality is helpful when multiple users are working on the same project You must be logged in to perform locking and unlocking.

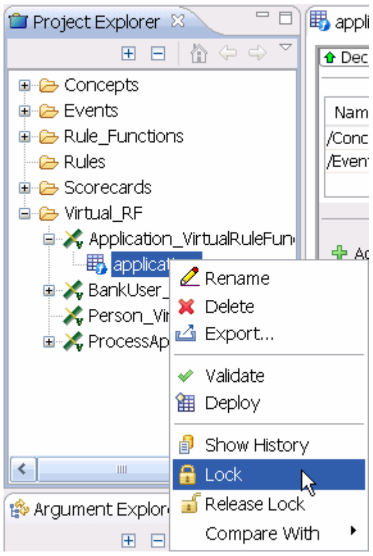
You must commit a newly created decision table at least once before you can use the lock option on it.

Locking a decision table does not stop other users from working on that table, but it does prevent them from committing it to the server. Other users cannot lock or release the lock on a table you have locked.

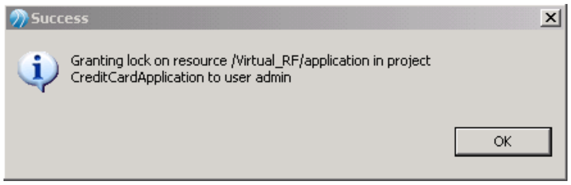
To release the lock, either explicitly release the lock, or commit the table. Committing a table releases the lock.

To Lock and Unlock a Decision Table

1. From the Project Explorer view, expand a VRF to view the desired decision table.
2. Right-click the decision table and select **Lock** or **Release Lock**, as appropriate.:

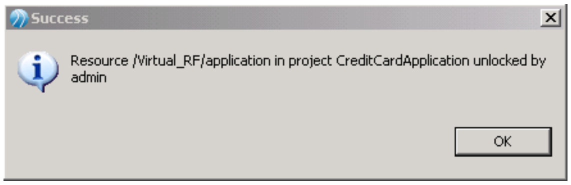


When you select Lock, a Success window displays with the locked resource name and the user's name who locked it.



Granting lock on resource /Virtual_RF/application in project CreditCardApplication to user admin

when you select Release Lock, a similar Success window displays:



Resource /Virtual_RF/application in project CreditCardApplication unlocked by admin

3. Click **OK**.

Validating Decision Tables

Validate the decision table before testing or committing it to RMS. To validate, from the Table menu, select **Validate**. You can also validate a decision table from Project Explorer. Right-click on the decision table and select **Validate**. If there are any access control violations or syntax errors in the tables, they are shown in the Problems View tab at the bottom of the application. Take any needed corrective actions and then validate the table again until all errors are resolved.

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

If a condition table cell is empty, it is skipped and thus considered to be evaluated as true.

Testing Decision Tables

When you have defined one or more decision tables, you can test them.

First configure for testing, then create test data, then deploy one or more decision tables, and lastly, test as shown in the task list below.

Task A Configure for Testing (as needed)

You can perform testing in a limited way (the default) or you can more closely mimic the production environment. For example you can enable channels and use the BusinessEvents project's object management options.

See Configuring Project Tester Options on page 31 for details.

Task B Create or Import Test Data

You can create data in the Test Data view, and you can export data for future use, and you can import data that was exported.

To Create Test Data

1. Open the decision project. Select `Project >` **Test**. An empty Test Data panel displays.
2. In Project Explorer, select an entity (concept or an event) for which you want to create test data. The editor panel displays test data (instances of that entity type) if any data is already defined.
3. Click **Add Data** to and enter test data for that instance. Repeat this step until you have created all the test instances of this entity you want to use for testing.

   Ensure that you enter valid data for each property's type. For example, if a property is of type Integer, enter a number and not a string.
4. Click **Save**.

   — Test data for a concept is saved to *DMProject*`/TestData/Concepts/`*ConceptName*`.testdata`.

   — Test data for an event is saved to *DMProject*`/TestData/Events/`*EventName*`.testdata`.
5. Repeat steps step 1 to step 4 until you have created all the test data you need.

To Export and Import Test Data

You export test data for each entity to separate comma-delimited files. The file format and naming convention is the same as that used by the test data Save operation: The extension `.concepttestdata` or `.eventtestdata` is appended to the name you provide.

You can then import that file into another project that can use the same test data.

- **To export**: With the desired entity's test data displayed in the Test Data view, click **`Export`**, browse to the desired directory and provide a filename.
- **To import**: With the desired entity displayed in the Test Data view, click **Import**, browse to the file and select it. If data exists for that entity, a confirmation window asks if you are sure you want to replace the data.

You can also use the file system to copy test data to another project's test data directory. Ensure that entity names in the target project match those you provide in the test data files.

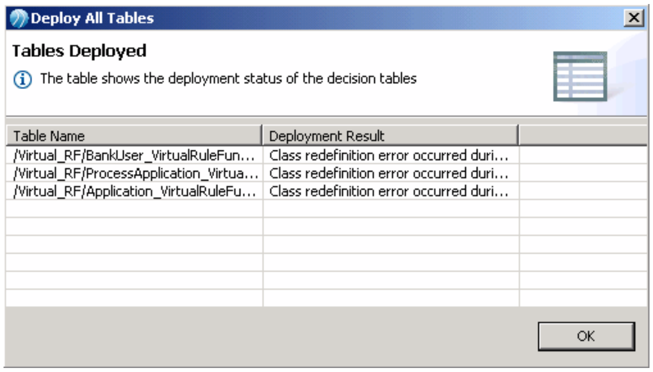
Task C Deploy and Test Decision Tables

This task is not required if you have checked the option "Automatically deploy all tables before starting project tests option. That option is in the Preferences > Testing panel. See Setting Application Preferences on page 108.

After creating or importing the test data, deploy the tables you want to test and then run the test. The selected decision tables are deployed to a BusinessEvents engine that runs inside Decision Manager for testing purposes.

1. As needed, open the decision project.
2. Do one of the following:
   — In Project Explorer, right click on a decision table and select **Deploy**. Repeat to deploy all desired decision tables.
   — Select **Project > Deploy Tables**.

Decision Manager validates the selected decision tables, compiles them, starts the BusinessEvents engine inside Decision Manager, and deploys the tables to the engine:



Deploy All Tables

Tables Deployed

The table shows the deployment status of the decision tables

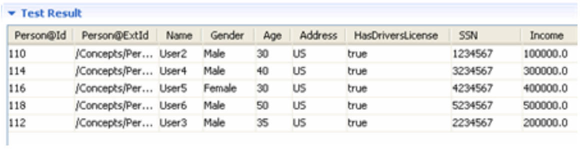
Table Name	Deployment Result
/Virtual_RF/BankUser_VirtualRuleFun...	Class redefinition error occurred duri...
/Virtual_RF/ProcessApplication_Virtua...	Class redefinition error occurred duri...
/Virtual_RF/Application_VirtualRuleFu...	Class redefinition error occurred duri...

3. Click **OK** to continue.

If any table fail to deploy, the engine stops and a dialog box lists all errors. The Console view also lists the errors.

4. In the Test Data viewer, click **Start Test**.

   The tester shows the values changed and the new instances created in Result Viewer, for example:

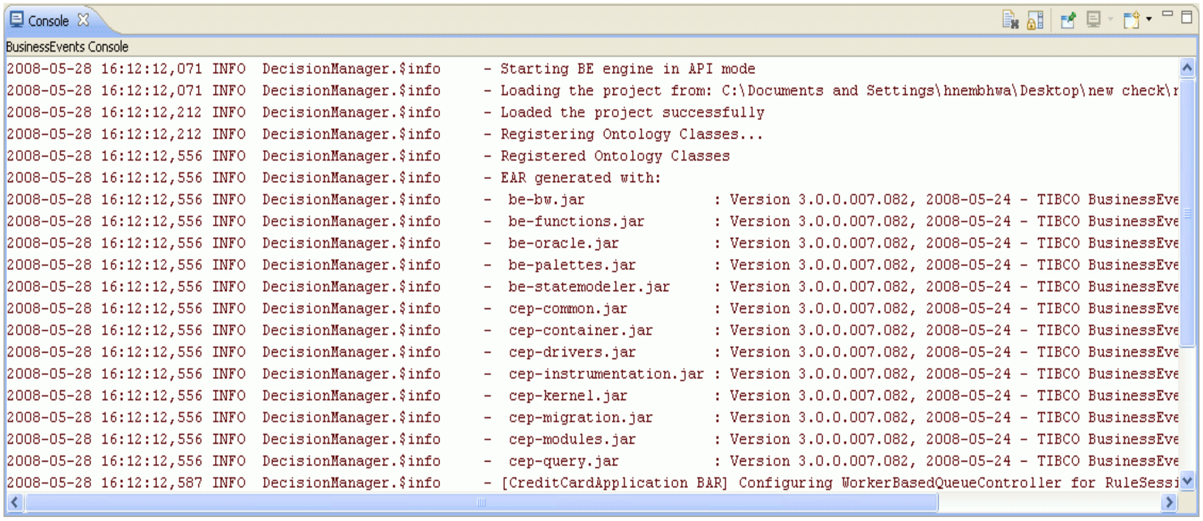


Test Result

Person@Id	Person@ExtId	Name	Gender	Age	Address	HasDriversLicense	SSN	Income
110	/Concepts/Per...	User2	Male	30	US	true	1234567	100000.0
114	/Concepts/Per...	User4	Male	40	US	true	3234567	300000.0
116	/Concepts/Per...	User5	Female	30	US	true	4234567	400000.0
118	/Concepts/Per...	User6	Male	50	US	true	5234567	500000.0
112	/Concepts/Per...	User3	Male	35	US	true	2234567	200000.0

5. The Console view shows log messages. For example, statements such as "System.debugOut ("Executed Rule")" are shown in this view. Click **Refresh** to

ensure you have the most up-to-date results, in case the rules have timer events.



```
2008-05-28 16:12:12,071 INFO  DecisionManager.$info    - Starting BE engine in API mode
2008-05-28 16:12:12,071 INFO  DecisionManager.$info    - Loading the project from: C:\Documents and Settings\hnembhwa\Desktop\new check\r
2008-05-28 16:12:12,212 INFO  DecisionManager.$info    - Loaded the project successfully
2008-05-28 16:12:12,212 INFO  DecisionManager.$info    - Registering Ontology Classes...
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    - Registered Ontology Classes
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    - EAR generated with:
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  be-bw.jar               : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  be-functions.jar        : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  be-oracle.jar           : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  be-palettes.jar         : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  be-statemodeler.jar     : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-common.jar          : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-container.jar       : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-drivers.jar         : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-instrumentation.jar : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-kernel.jar          : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-migration.jar       : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-modules.jar         : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,556 INFO  DecisionManager.$info    -  cep-query.jar           : Version 3.0.0.007.082, 2008-05-24 - TIBCO BusinessEve
2008-05-28 16:12:12,587 INFO  DecisionManager.$info    - [CreditCardApplication BAR] Configuring WorkerBasedQueueController for RuleSessi
```

6. Click **Stop Test** to reset the tester rule session.
7. Click **Shutdown Engine** to stop all rule sessions and stop the engine.
8. Click **Project > Undeploy Tables** to remove all the decision tables from the engine so you can deploy individual tables.

Committing the Project for Approval

When your decision tables are working as desired, you can commit them for approval, and eventual deployment to the BusinessEvents application.

See Committing Decision Tables for Approval on page 61 for details.

Comparing Decision Tables

You can compare the structure of different decision tables.

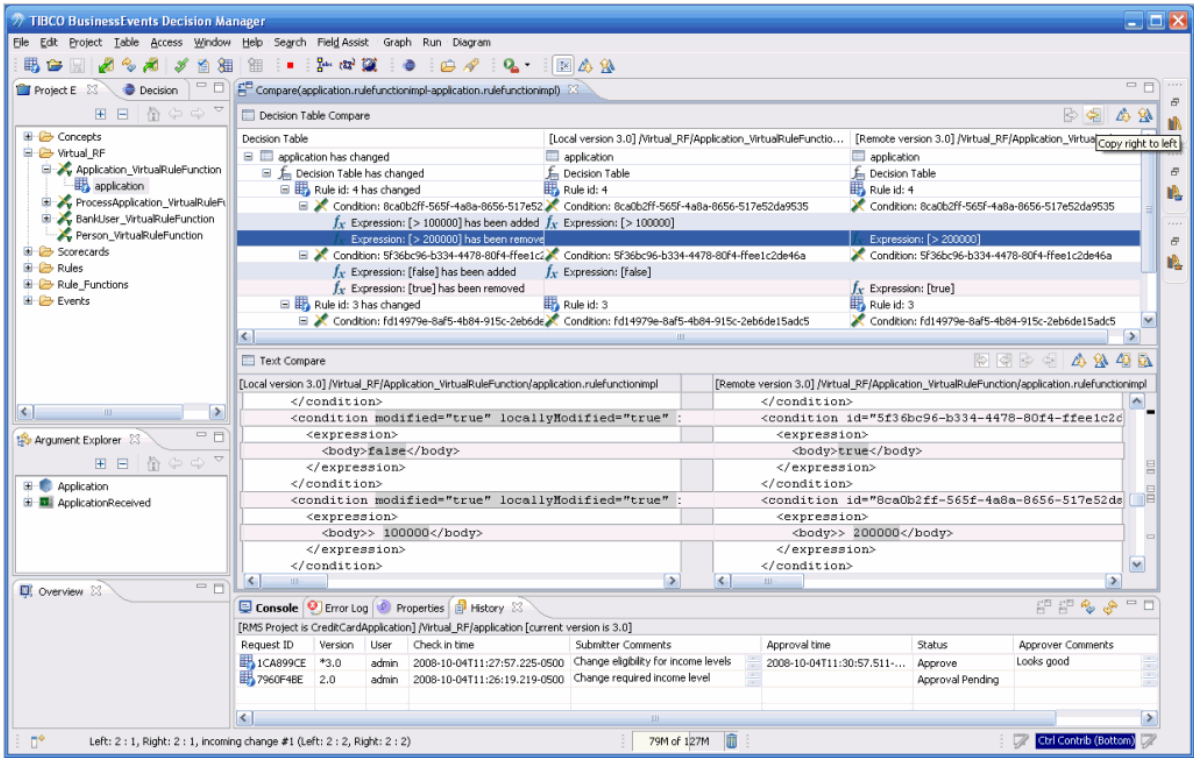
You can compare two local decision tables by selecting two tables from the Project Explorer and selecting **Compare With > Each other** from the context menu.

Alternatively, a local decision table can be compared with the latest approved version by selecting a single decision table in the Project Explorer and selecting **Compare With > Approved Copy**.

It is also possible to compare two arbitrary decision table versions. See Working With History View, page 87 for more details.

By default, opening the compare editor will show structural as well as textual differences of a decision table without opening the decision table editor. Any additions, deletions, or changes to a decision table are shown. This provides the user with a high level overview of what has changed between versions of a decision table, and could be useful during the commit or approval process.

From the structural compare window, it is possible to merge changes in a decision table by selecting individual differences and clicking the **Copy** actions on the top right of the structural compare window. For instance, to overwrite a local change with the remote version, select the change and click **Copy right to left**. This can be useful when the approved version of the decision table is newer than the local version, but the local version has been modified and should not be overwritten via the Update command. It is not possible to merge remote decision table versions, and the Copy actions are disabled when remote versions are compared.


TIBCO BusinessEvents Decision Manager
File Edit Project Table Access Window Help Search Field Assist Graph Run Diagram
Project E
Decision
Concepts
Virtual_RF
Application_VirtualRuleFunction
application
ProcessApplication_VirtualRuleFu
BankUser_VirtualRuleFunction
Person_VirtualRuleFunction
Scorecards
Rules
Rule_Functions
Events
Argument Explorer
Application
ApplicationReceived
Overview
Compare(application.rulefunctionimpl-application.rulefunctionimpl)
Decision Table Compare
Copy right to left
Text Compare
Console
Error Log
Properties
History
79M of 127M
Ctrl Contrib (Bottom)

Decision Table	[Local version 3.0] /Virtual_RF/Application_VirtualRuleFunctio...	[Remote version 3.0] /Virtual_RF/Application_Virtua
application has changed	application	application
Decision Table has changed	Decision Table	Decision Table
Rule id: 4 has changed	Rule id: 4	Rule id: 4
Condition: 8ca0b2ff-565f-4a8a-8656-517e52	Condition: 8ca0b2ff-565f-4a8a-8656-517e52da9535	Condition: 8ca0b2ff-565f-4a8a-8656-517e52da9535
Expression: [> 100000] has been added	Expression: [> 100000]	
Expression: [> 200000] has been remove		Expression: [> 200000]
Condition: 5f36bc96-b334-4478-80f4-ffee1c2	Condition: 5f36bc96-b334-4478-80f4-ffee1c2de46a	Condition: 5f36bc96-b334-4478-80f4-ffee1c2de46a
Expression: [false] has been added	Expression: [false]	
Expression: [true] has been removed		Expression: [true]
Rule id: 3 has changed	Rule id: 3	Rule id: 3
Condition: fd14979e-8af5-4b84-915c-2eb6de	Condition: fd14979e-8af5-4b84-915c-2eb6de15adc5	Condition: fd14979e-8af5-4b84-915c-2eb6de15adc5

[Local version 3.0] /Virtual_RF/Application_VirtualRuleFunction/application.rulefunctionimpl

```
    </condition>
    <condition modified="true" locallyModified="true" :
      <expression>
        <body>false</body>
      </expression>
    </condition>
    <condition modified="true" locallyModified="true" :
      <expression>
        <body>> 100000</body>
      </expression>
    </condition>
```

[Remote version 3.0] /Virtual_RF/Application_VirtualRuleFunction/application.rulefunctionimpl

```
    </condition>
    <condition id="5f36bc96-b334-4478-80f4-ffee1c2d
      <expression>
        <body>true</body>
      </expression>
    </condition>
    <condition id="8ca0b2ff-565f-4a8a-8656-517e52da
      <expression>
        <body>> 200000</body>
      </expression>
    </condition>
```

[RMS Project is CreditCardApplication] /Virtual_RF/application [current version is 3.0]

Request ID	Version	User	Check in time	Submitter Comments	Approval time	Status	Approver Comments
1CA899CE	*3.0	admin	2008-10-04T11:27:57.225-0500	Change eligibility for income levels	2008-10-04T11:30:57.511-...	Approve	Looks good
7960F4BE	2.0	admin	2008-10-04T11:26:19.219-0500	Change required income level		Approval Pending	

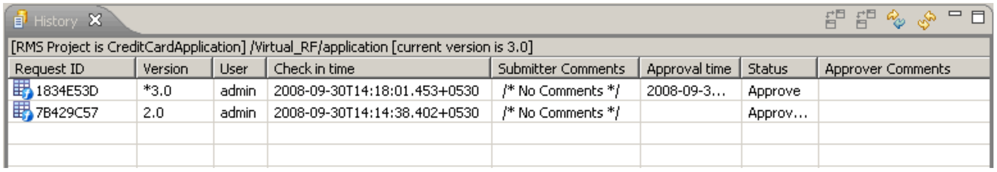
Left: 2 : 1, Right: 2 : 1, incoming change #1 (Left: 2 : 2, Right: 2 : 2)

Working With History View

To view history, right click on a decision table in the Project Explorer and select Show History

Columns in the History view are as follows:

- **Request ID**: This displays the unique request ID that is sent upon successful check-in operation. This check-in operation also contains the said resource.
- **Version**: The version number of the resource at the time of the check-in operation.
- **User**: The username who committed this resource.
- **Checkin Time**: UTC time of check-in.
- **Submitter Comments**: Optional comments added by user during check-in.
- **Approval Time**: The UTC time of approval of the resource.
- **Status**: Shows the status of the commit, as also seen in the Worklist or My Status views.
- **Approver Comments**: Optional comments added by the Approver during approval process.



[RMS Project is CreditCardApplication] /Virtual_RF/application [current version is 3.0]

Request ID	Version	User	Check in time	Submitter Comments	Approval time	Status	Approver Comments
1834E53D	*3.0	admin	2008-09-30T14:18:01.453+0530	/* No Comments */	2008-09-3...	Approve	
7B429C57	2.0	admin	2008-09-30T14:14:38.402+0530	/* No Comments */		Approv...	

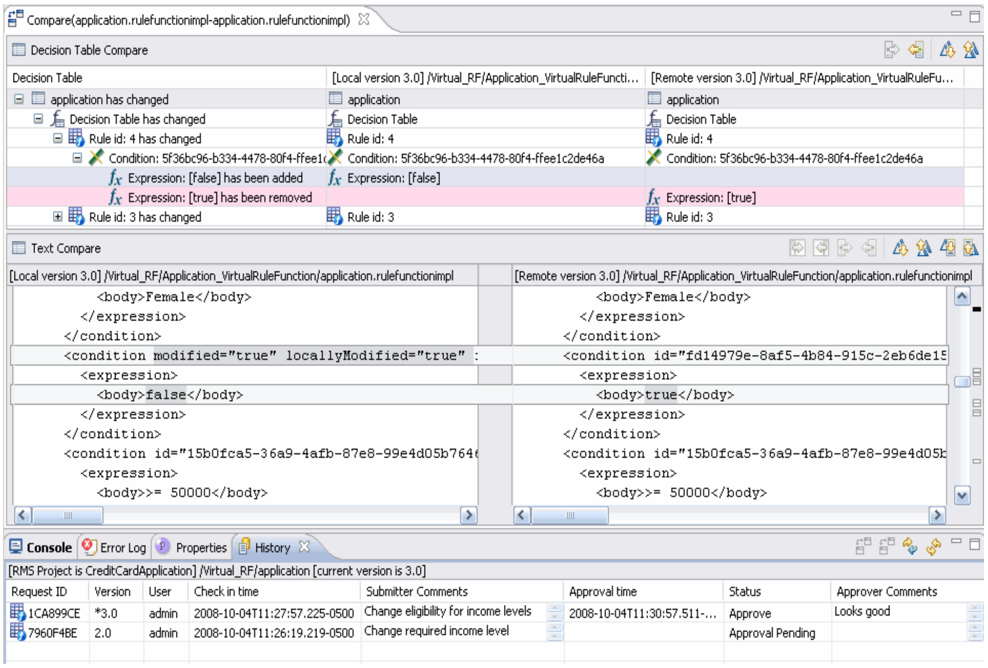
If a row selected in the history view is double clicked, it opens the checked-in resource in the decision table editor. This is used as a means of review either by a reviewer, or even by another decision table author.

There are a few context menu options available upon selection of a row:

- **Compare With Current version**: Compares the selected check-in with the one currently in the loaded project. (Note that a decision project is already open with the resource in it.)
- **Open in Editor**: Same operation as double click.
- **Update**: Updates the local resource with the latest version from RMS.
- **Refresh**: Refresh the History view for any check-ins made on the resource after the first time the view was invoked.

Comparing Two Checkin Requests

The history view can also be used as a means of comparing two checkin requests. If two requests (rows in the view) are selected, the context menu provides an option to open the Compare editor. See Comparing Decision Tables on page 85.



Compare(application.rulefunctionimpl-application.rulefunctionimpl)

Decision Table Compare

Decision Table	[Local version 3.0] /Virtual_RF/Application_VirtualRuleFuncti...	[Remote version 3.0] /Virtual_RF/Application_VirtualRuleFu...
application has changed	application	application
Decision Table has changed	Decision Table	Decision Table
Rule id: 4 has changed	Rule id: 4	Rule id: 4
Condition: 5f36bc96-b334-4478-80f4-ffee1c	Condition: 5f36bc96-b334-4478-80f4-ffee1c2de46a	Condition: 5f36bc96-b334-4478-80f4-ffee1c2de46a
Expression: [false] has been added	Expression: [false]	
Expression: [true] has been removed		Expression: [true]
Rule id: 3 has changed	Rule id: 3	Rule id: 3

Text Compare

[Local version 3.0] /Virtual_RF/Application_VirtualRuleFunction/application.rulefunctionimpl

```
          <body>Female</body>
        </expression>
      </condition>
      <condition modified="true" locallyModified="true" :
        <expression>
          <body>false</body>
        </expression>
      </condition>
      <condition id="15b0fca5-36a9-4afb-87e8-99e4d05b764
        <expression>
          <body>>= 50000</body>
```

[Remote version 3.0] /Virtual_RF/Application_VirtualRuleFunction/application.rulefunctionimpl

```
          <body>Female</body>
        </expression>
      </condition>
      <condition id="fd14979e-8af5-4b84-915c-2eb6de15
        <expression>
          <body>true</body>
        </expression>
      </condition>
      <condition id="15b0fca5-36a9-4afb-87e8-99e4d05b
        <expression>
          <body>>= 50000</body>
```

Console | Error Log | Properties | History

[RMS Project is CreditCardApplication] /Virtual_RF/application [current version is 3.0]

Request ID	Version	User	Check in time	Submitter Comments	Approval time	Status	Approver Comments
1CA899CE	*3.0	admin	2008-10-04T11:27:57.225-0500	Change eligibility for income levels	2008-10-04T11:30:57.511-...	Approve	Looks good
7960F4BE	2.0	admin	2008-10-04T11:26:19.219-0500	Change required income level		Approval Pending	

Importing and Exporting Decision Tables

This section explains how to import data from a Microsoft Excel file and export a decision table to a Microsoft Excel file.

Importing Data from a Microsoft Excel File

You can import data from a Microsoft Excel file to create a decision table. For some examples see *BE_HOME*`/DecisionManager/examples/ExcelFiles`.

To Import Data

1. Open the decision project.
2. You can import data from the File menu or Project Explorer panel.

- From the **File** menu:
  - — Select **Import**. An Import dialog is displayed.
  - — Click **Browse** and select the Microsoft Excel file to import
  - — Enter the name of decision table.
  - — Select the VRF you want to implement with the decision table, and click **OK**.
- From **Project Explorer**:
  - — Select a VRF to import data for a decision table, from the VRF list.
  - — Right-click on the selected VRF and select **Import**.
  - — Import dialog is displayed with VRF selected. Select the Microsoft Excel file to import and enter the name of the decision table.

Importing data from Project Explorer:

- Is recommended when there is a large list of projects.
- Does not allow you to choose any other source (VRF) available in the tree other than the selected.

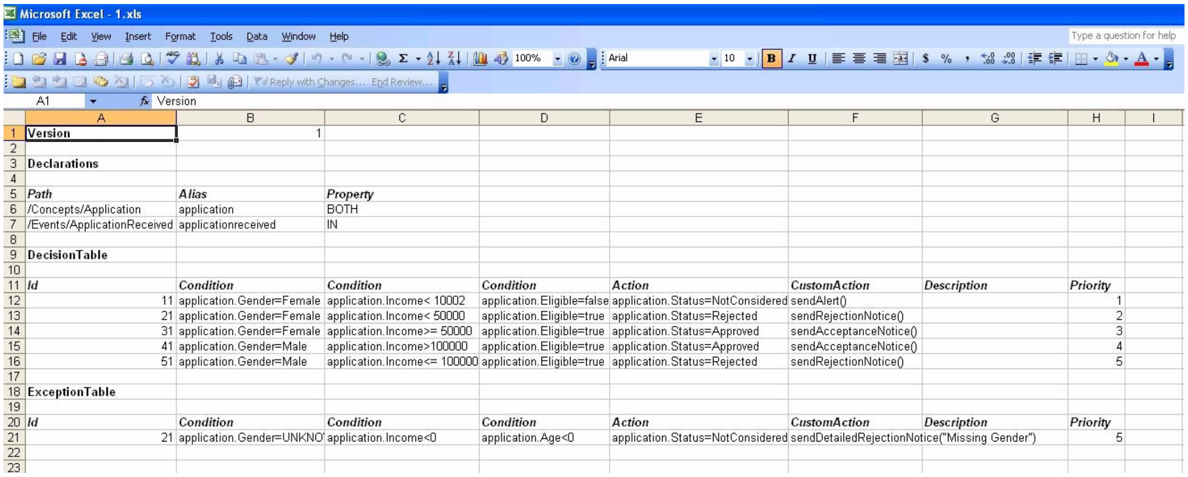
The data imported from the Microsoft Excel file is saved to the decision table and you can continue modifying the data or add new data.

For custom conditions and custom actions to be imported from Excel, mark the header names as "CustomCondition" and "CustomAction" respectively.

Although row descriptions are imported, condition and action specific comments are not imported.

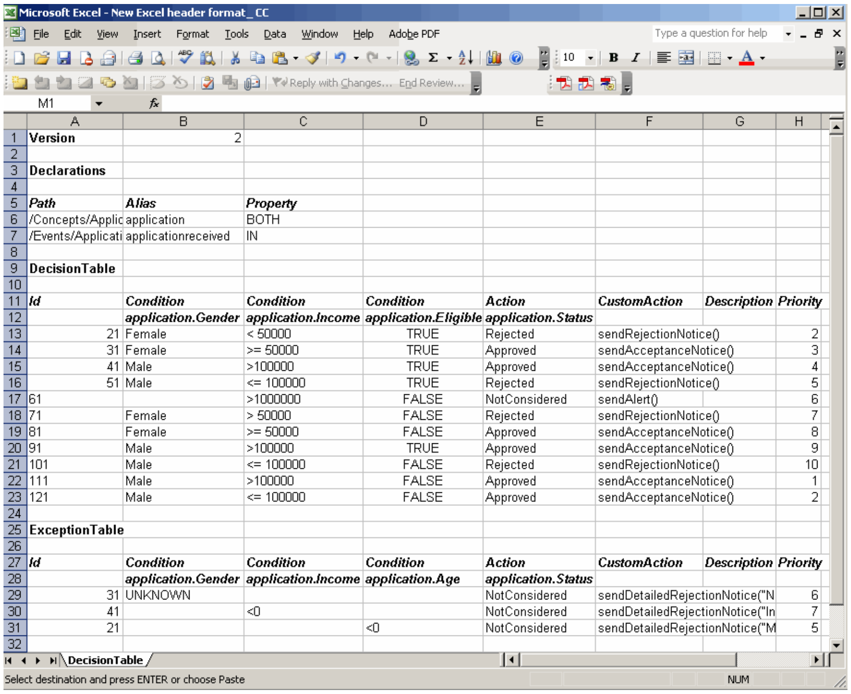
Decision Manager supports two formats, an older one and the current one. The current format requires Version header. While importing using the older format, a Version header in the Excel file is not required.

Figure 1 Microsoft Excel File Format - Old


Microsoft Excel - 1.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Reply with Changes... End Review...
A1 Version

	A	B	C	D	E	F	G	H	I
1	**Version**	1							
2									
3	**Declarations**								
4									
5	***Path***	***Alias***	***Property***						
6	/Concepts/Application	application	BOTH						
7	/Events/ApplicationReceived	applicationreceived	IN						
8									
9	**DecisionTable**								
10									
11	***Id***	***Condition***	***Condition***	***Condition***	***Action***	***CustomAction***	***Description***	***Priority***	
12	11	application.Gender=Female	application.Income< 10002	application.Eligible=false	application.Status=NotConsidered	sendAlert()		1	
13	21	application.Gender=Female	application.Income< 50000	application.Eligible=true	application.Status=Rejected	sendRejectionNotice()		2	
14	31	application.Gender=Female	application.Income>= 50000	application.Eligible=true	application.Status=Approved	sendAcceptanceNotice()		3	
15	41	application.Gender=Male	application.Income>100000	application.Eligible=true	application.Status=Approved	sendAcceptanceNotice()		4	
16	51	application.Gender=Male	application.Income<= 100000	application.Eligible=true	application.Status=Rejected	sendRejectionNotice()		5	
17									
18	**ExceptionTable**								
19									
20	***Id***	***Condition***	***Condition***	***Condition***	***Action***	***CustomAction***	***Description***	***Priority***	
21	21	application.Gender=UNKNO'	application.Income<0	application.Age<0	application.Status=NotConsidered	sendDetailedRejectionNotice("Missing Gender")		5	
22									
23									

Figure 2 Microsoft Excel File Format - New


Microsoft Excel - New Excel header format_CC
File Edit View Insert Format Tools Data Window Help Adobe PDF
Type a question for help
Reply with Changes... End Review...
M1
DecisionTable
Select destination and press ENTER or choose Paste
NUM

	A	B	C	D	E	F	G	H
1	**Version**	2						
2								
3	**Declarations**							
4								
5	***Path***	***Alias***	***Property***					
6	/Concepts/Applic	application	BOTH					
7	/Events/Applicati	applicationreceived	IN					
8								
9	**DecisionTable**							
10								
11	***Id***	***Condition***	***Condition***	***Condition***	***Action***	***CustomAction***	***Description***	***Priority***
12		***application.Gender***	***application.Income***	***application.Eligible***	***application.Status***			
13	21	Female	< 50000	TRUE	Rejected	sendRejectionNotice()		2
14	31	Female	>= 50000	TRUE	Approved	sendAcceptanceNotice()		3
15	41	Male	>100000	TRUE	Approved	sendAcceptanceNotice()		4
16	51	Male	<= 100000	TRUE	Rejected	sendRejectionNotice()		5
17	61		>1000000	FALSE	NotConsidered	sendAlert()		6
18	71	Female	> 50000	FALSE	Rejected	sendRejectionNotice()		7
19	81	Female	>= 50000	FALSE	Approved	sendAcceptanceNotice()		8
20	91	Male	>100000	TRUE	Approved	sendAcceptanceNotice()		9
21	101	Male	<= 100000	FALSE	Rejected	sendRejectionNotice()		10
22	111	Male	>100000	FALSE	Approved	sendAcceptanceNotice()		1
23	121	Male	<= 100000	FALSE	Approved	sendAcceptanceNotice()		2
24								
25	**ExceptionTable**							
26								
27	***Id***	***Condition***	***Condition***	***Condition***	***Action***	***CustomAction***	***Description***	***Priority***
28		***application.Gender***	***application.Income***	***application.Age***	***application.Status***			
29	31	UNKNOWN			NotConsidered	sendDetailedRejectionNotice("N		6
30	41		<0		NotConsidered	sendDetailedRejectionNotice("In		7
31	21			<0	NotConsidered	sendDetailedRejectionNotice("M		5
32								

The new Excel file format requires Version header and version number.

In the new Excel file format:

- The Condition and Action column contain their respective body in the first row followed by their values.
- The CustomAction, CustomCondition, Description, ID, and Priority rows are blank (no body) followed by their actual values.

Exporting a Decision Table to a Microsoft Excel File

You can export the decision table to a Microsoft Excel file. To export:

1. Open the decision project.
2. You can export the decision table from the File menu or Project Explorer panel.

- From the File menu:
  - — Select **Export**. An Export dialog is displayed
  - — Click **Browse**
  - — Select the name and path to save the file as an Excel file, and click **OK.**
- From Project Explorer:
  - — Select a decision table from the VRF list to export.
  - — Right-click on the selected decision table and select Export.

Not Exported A priority setting of 5 in a decision table row (because 5 is the default value).

The decision table is now exported in the new header format with a header containing version and version number. See the above figure for the new Excel file format.

Chapter 8 Working with the Approval Process

This chapter explains the approval workflow

Topics

Working with the Decision Table Approval Process

Business users create, modify, and delete decision tables, test data, and domain models as appropriate. Then they submit their work for approval, as explained in Committing Decision Tables for Approval on page 61.

Projects that business users have committed for approval appear in the worklist of users with the role of `RULE_ADMINISTRATOR` (and any other role configured with this permission). Approvers can approve the project or reject it

See Checking the Status of a Committed Project on page 62 for the related business user procedure.

Also in the Worklist area is a status that unloads (undeploys) a deployed decision table from a running BusinessEvents engine.

This section is written for users with permissions to perform project approval.

This manual documents the behavior of the product as shipped. The workflow, the roles used, and the permissions granted to the roles are all configurable. However, the general flow is likely to be similar to that described in this guide.

See Decision Manager and RMS Workflow on page 7 for an overview and see Chapter 2, RMS and Decision Manager Configuration, page 13 for details about configuration and customization.

Decision Tables

You assign an approval status to each individual commit request for a project. A commit request can contain zero or more decision tables. Most of the content of this chapter deals with the approval process for decision tables.

Test Data

Test data in approved Decision Manager projects is not automatically made available to others. It is considered to be local to each user's project. However, you can manually copy test data to the RMS project as desired. It is then made available to other Decision Manager users when they check out the project.

Copy the data files and subdirectories from the *DecisionProject*`/data` to the *RMSProject*`/data` directory.

See Testing Decision Tables on page 81 for more details.

Domain Model File

An approved domain model file automatically overwrites the RMS project domain model file. Domain model files are also versioned, similar to all decision table files.

The domain model file itself is not visible for separate approval. When a user commits a decision table for approval, any domain model changes made (if any) are also committed. Upon approval, the updated domain model replaces the master copy in RMS.

Understanding Approval Actions and Status Values

When a user commits one or more resources for approval, the status of the request is set to Approval Pending. The approver then sets the status as appropriate.

You can set the Action field on a decision project resource to the following values:

Approve

When you approve a project for deployment, the following actions occur:

- Resources are checked in to the RMS project and are available for Decision Manager users to check out.
- Class files for the approved decision tables are generated in the `deployment` directory (or whatever directory is configured. See Configuring RMS Server Properties on page 16.)

When you approve a project for deletion, the following actions occur:

- RMS deletes the master copy of the table.
- The generated class file for the table is deleted.

Reject

Informs the business user that the change has been rejected. It is the business user's responsibility to take appropriate action, such as deleting their local copy.

Unload

This option applies only to decision tables that have been deployed to the BusinessEvents production system and whose current status is Approve (or Approval Pending, after a failed unload).

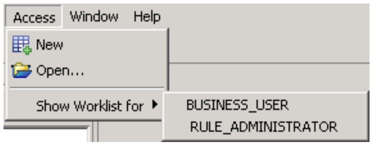
When you choose unload RMS undeploys the table from a production BusinessEvents system. This action can only be done when the BusinessEvents system is running.

If unload fails, the table's status returns to Approval Pending. From that status you can again attempt to set the Action to Unload. See Chapter 9, Deploying and Unloading Decision Tables, on page 101 for more details.

Checking a Worklist and Taking Action on Project Resources

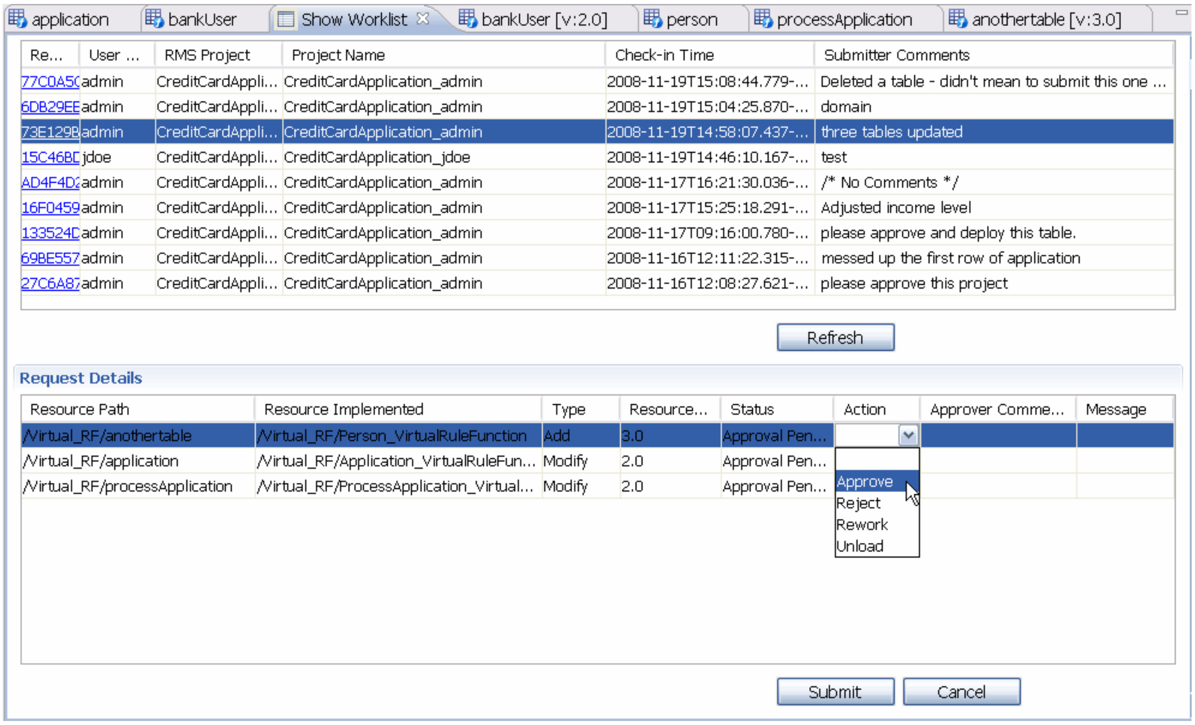
To Check Your Worklist

1. From the Access menu, select **Show Worklist for** and select one of the roles. Roles displayed have permission to approve projects.



   Decision Manager connects to RMS and displays all the requests submitted for approval.

2. Click a request ID in the upper panel to display details in the Request Details panel below.



Re...	User ...	RMS Project	Project Name	Check-in Time	Submitter Comments
77C0A5C	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-19T15:08:44.779-...	Deleted a table - didn't mean to submit this one ...
6DB29EE	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-19T15:04:25.870-...	domain
73E129B	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-19T14:58:07.437-...	three tables updated
15C46BD	jdoe	CreditCardAppli...	CreditCardApplication_jdoe	2008-11-19T14:46:10.167-...	test
AD4F4D2	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-17T16:21:30.036-...	/* No Comments */
16F0459	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-17T15:25:18.291-...	Adjusted income level
133524D	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-17T09:16:00.780-...	please approve and deploy this table.
69BE557	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-16T12:11:22.315-...	messed up the first row of application
27C6A87	admin	CreditCardAppli...	CreditCardApplication_admin	2008-11-16T12:08:27.621-...	please approve this project

Resource Path	Resource Implemented	Type	Resource...	Status	Action	Approver Comme...	Message
/Virtual_RF/anothertable	/Virtual_RF/Person_VirtualRuleFunction	Add	3.0	Approval Pen...			
/Virtual_RF/application	/Virtual_RF/Application_VirtualRuleFun...	Modify	2.0	Approval Pen...			
/Virtual_RF/processApplication	/Virtual_RF/ProcessApplication_Virtual...	Modify	2.0	Approval Pen...			

3. To examine the decision table before taking an action, double-click a resource in the Request Details panel. the decision table displays in another view.

4. In the Action field of the Request Details panel, select a value:

   — Approve

   — Reject:

   — Unload

   See Understanding Approval Actions and Status Values on page 95 for more details.

5. Click **Submit**. The Status fields display the status you chose in the Action field.

Checking in Decision Tables Using JMX

Upon approval of a checkin request, RMS copies approved decision table files into the RMS project `decisiondata` directory. However, you can't just manually copy decision tables into that directory. Either you must follow the commit and approval process, or you can perform an explicit check-in using JMX. Only if you do one of these procedures are decision tables available for Decision Manager users to check out.

Explicit check-in using JMX has two options.

- The **`checkinExtEntity`** option checks in the named decision project artifact.
- The **`checkinExtEntities`** option checks in all decision project artifacts (such as decision tables and domain model files) that are not yet checked in.

This procedure makes decision tables available for Decision Manager users. It does not create the class files required for deployment. See Deploying Decision Tables on page 102 for more details.

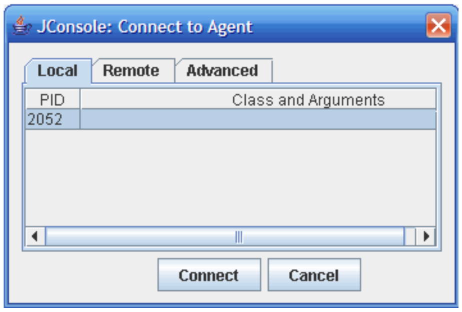
Configure be-rms.tra With JMX Properties and Start the Server

Ensure that the following properties are present in the *BE_HOME*`/rms/bin/be-rms.tra` file and restart the server:

```
java.property.com.sun.management.jmxremote=true
java.property.com.sun.management.jmxremote.ssl=false
java.property.com.sun.management.jmxremote.port=5561
java.property.com.sun.management.jmxremote.authenticate=false
```

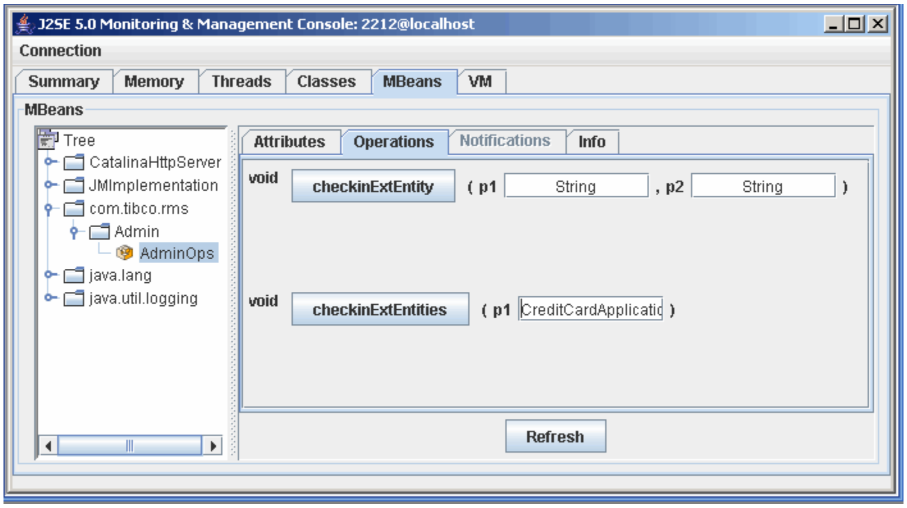
Check in Decision Tables

1. Open a command window in the bin directory of your JDK installation and type **`jconsole`**. You see a dialog box similar to the following:



2. Select the connection corresponding to the node that you want to view. (Each node runs in one JVM.) Click Connect.

3. Select the MBeans tab and in the left panel, navigate to `com.tibco.rms` > `Admin` > `AdminOps`.
4. In the right panel, select the **Operations** tab.



5. Do one of the following.

   To check in a specified decision project artifact:

   a. For the first parameter, specify the name of the RMS project. It must be in the correctly configured location (see Configuring RMS Server Properties on page 16). For example, if you were working with the example credit card application RMS project, you would enter `CreditCardApplication`.

   b. For the second parameter specify the full path to the artifact. For example:

   ```
   C:\tibco\be\3.0\rms\examples\CreditCardApplication\decisiond
   ata\_Virtual_RF_BankUser_VirtualRuleFunction_bankUser.rulefu
   nctionimpl
   ```

   c. Click **`checkinExtEntity`**.

   To check in all decision project artifacts that are not yet checked in:

   a. Specify the name of the RMS project. It must be in the correctly configured location (see Configuring RMS Server Properties on page 16). For example, if you were working with the example credit card application RMS project, you would enter `CreditCardApplication`.

   b. Click **`checkinExtEntities`**.

Chapter 9 Deploying and Unloading Decision Tables

After decision tables are approved, their Java classes are read for deployment to the BusinessEvents application. Deployed tables can also be unloaded from the BusinessEvents application.

See Configuring for Deployment of Decision Table Classes on page 26 and Configuring for Unloading of Decision Table Classes on page 29 for configuration steps required before you can deploy decision tables.

Topics

Deploying Decision Tables

Configuration required Ensure that RMS and BusinessEvents are configured for deployment of decision table classes before you begin. See Configuring for Deployment of Decision Table Classes on page 26 and Configuring for Unloading of Decision Table Classes on page 29 for details.

When decision project resources are approved, RMS generates the decision table class files and places them in the `deployment` subdirectory of the RMS project directory. (This location is configurable. See note above.)

You can now copy the files to the directory where BusinessEvents expects to find Decision Manager decision table class files.

You can configure the corresponding BusinessEvents property `be.engine.cluster.externalClasses.path` to point to the `deployment` subdirectory of one RMS project. Then when decision tables are approved for that project they are automatically available for deployment to BusinessEvents without manual copying. See Virtual Rule Functions and Decision Manager in *TIBCO BusinessEvents User's Guide* for more about BusinessEvents configuration.

Deploying Decision Table Classes at BusinessEvents Startup

To deploy decision table classes when the BusinessEvents production system starts up, a technical user copies them to a location known to the BusinessEvents cluster nodes. See Configuring for Deployment of Decision Table Classes on page 26 to determine the location configured for your system.

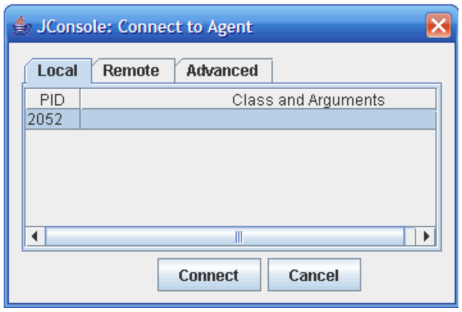
Because these classes originate in the Decision Manager application, they are known as *external classes*, from the point of view of the BusinessEvents nodes.

Hot Deploying External Classes Using JMX

If RMS is configured with JMX properties, you can hot deploy external classes to BusinessEvents cluster nodes. For JMX configuration details, see Configuring for Deployment of Decision Table Classes on page 26.

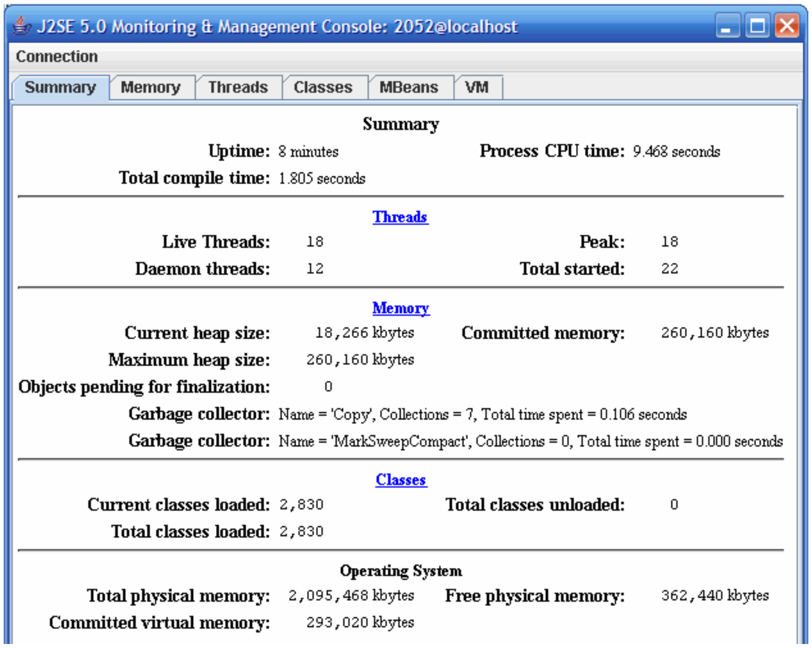
To Hot Deploy External Classes using JMX

1. Open a command window in the bin directory of your JDK installation and type `jconsole`. You see a dialog box similar to the following:



PID	Class and Arguments
2052	

2. Select the connection corresponding to the BusinessEvents node that you want to view. (Each node runs in one JVM.) Click **Connect**. You see a dialog box similar to the following:



Summary

Uptime: 8 minutes	Process CPU time: 9.468 seconds
Total compile time: 1.805 seconds

Threads

Live Threads: 18	Peak: 18
Daemon threads: 12	Total started: 22

Memory

Current heap size: 18,266 kbytes	Committed memory: 260,160 kbytes
Maximum heap size: 260,160 kbytes
Objects pending for finalization: 0
Garbage collector: Name = 'Copy', Collections = 7, Total time spent = 0.106 seconds
Garbage collector: Name = 'MarkSweepCompact', Collections = 0, Total time spent = 0.000 seconds

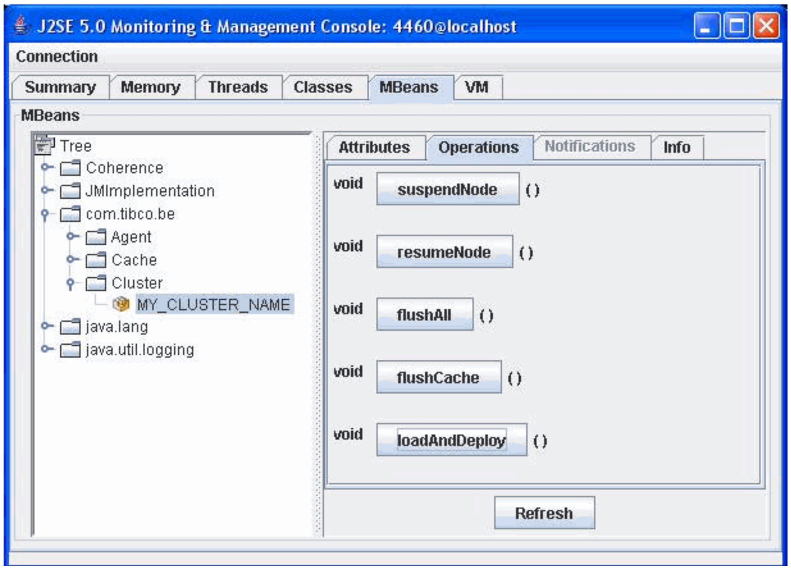
Classes

Current classes loaded: 2,830	Total classes unloaded: 0
Total classes loaded: 2,830

Operating System

Total physical memory: 2,095,468 kbytes	Free physical memory: 362,440 kbytes
Committed virtual memory: 293,020 kbytes

3. Select the **MBeans** tab and expand the nodes under Tree in the left panel. You see a dialog box similar to the following:



4. Navigate to the `com.tibco.be` tree, expand `Cluster`, and select the appropriate cluster name.
5. In the right panel, select the **Operations** tab and click **loadAndDeploy**.

The class files generated from decision tables are deployed to all nodes in the cluster.

Undeploying (Unloading) Decision Tables

You can unload a decision table from a running BusinessEvents production system. The system pauses briefly until the procedure is complete.

You might want to do this, for example if you want to replace one VRF implementation (decision table) with a different one.

Undeploying Decision Manager Classes in Decision Manager

1. Log in to Decision Manager and select **Access > Show Worklist for**.
2. Select a deployed decision table and select **Unload**.

Undeploying Decision Manager Classes Using JMX

Follow all instructions in Hot Deploying External Classes Using JMX on page 103.

Replace step 5 with the following:

- In the right panel, select the **Operations** tab. Specify the URI of the virtual rule function and the decision table name and click **unloadClass**. The corresponding class is unloaded.

Chapter 10 Setting Application Preferences

This chapter briefly explains how to set application preferences.

Topics

Setting Application Preferences

You can change the appearance of Decision Manager by setting your own preferences. You can set preferences for General options, Console View, Decision Tables, Compare editor, Problems View, and Testing. For decision tables, you can change table view, editor menu options, condition and action field filters, advanced settings, colors, and fonts.

To Set Preferences:

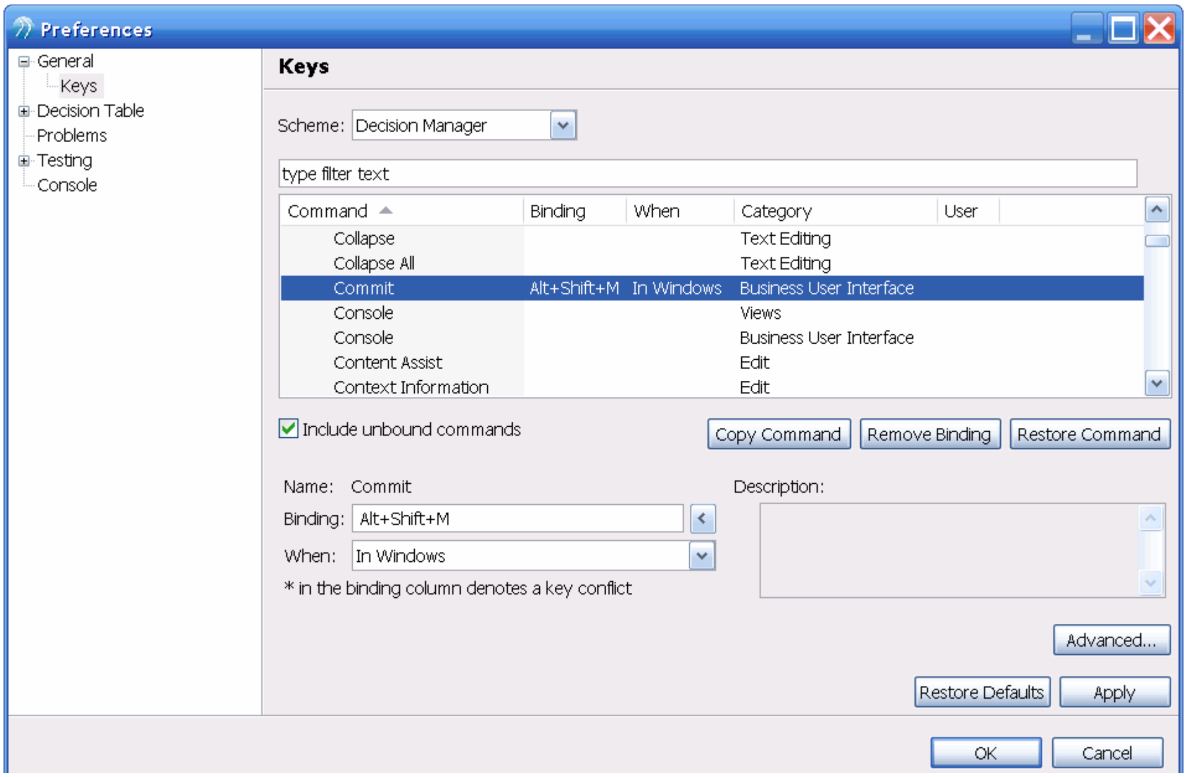
1. From the Window menu, select **Preferences**.
2. Expand the options on the left and select each option to view a panel of options on the right.

   Details on each set of options are provided in sections following.
3. Change the options from the list as desired, and then click **Apply**.
4. Click **OK** to see Decision Manager with your preferred settings.

General Preferences

Set options to show memory usage status, and to allow rule viewing as desired.

General Preferences—Keys



Scheme: Decision Manager

type filter text

Command	Binding	When	Category	User
Collapse			Text Editing	
Collapse All			Text Editing	
Commit	Alt+Shift+M	In Windows	Business User Interface	
Console			Views	
Console			Business User Interface	
Content Assist			Edit	
Context Information			Edit	

[✓] Include unbound commands

Name: Commit

Binding: Alt+Shift+M

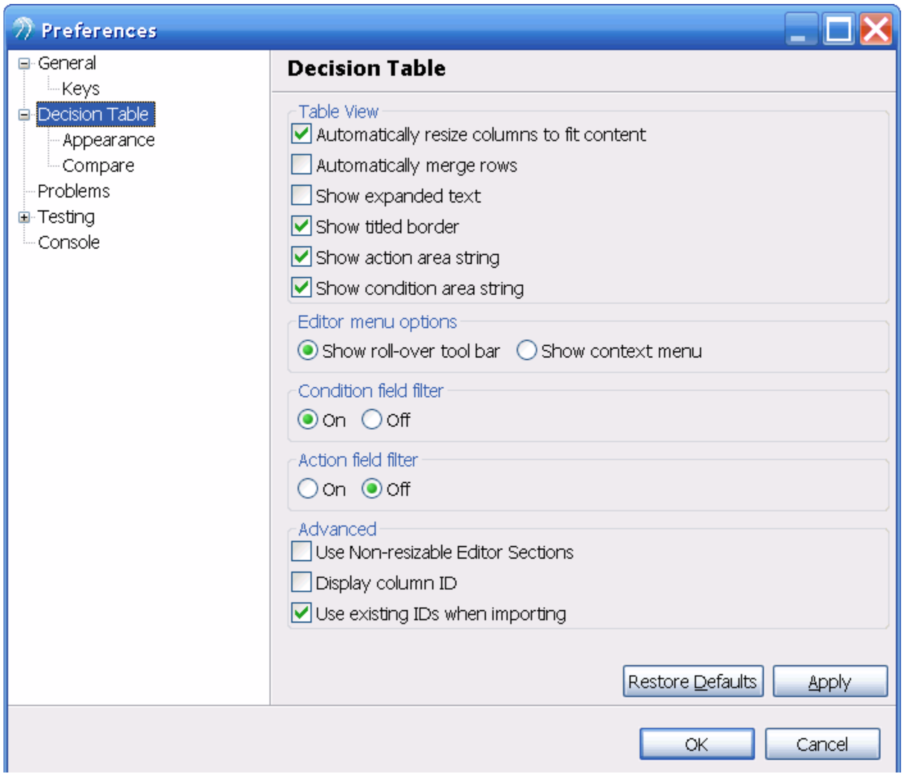
When: In Windows

* in the binding column denotes a key conflict

Description:

Using the Keys dialog, you can create keyboard shortcuts to jump to different areas of the user interface. You can also edit existing shortcuts to suit your needs.

Decision Table Preferences



Decision Table

Table View

- [✓] Automatically resize columns to fit content
- [ ] Automatically merge rows
- [ ] Show expanded text
- [✓] Show titled border
- [✓] Show action area string
- [✓] Show condition area string

Editor menu options

[✓ Show roll-over tool bar] Show context menu

Condition field filter

[✓ On] Off

Action field filter

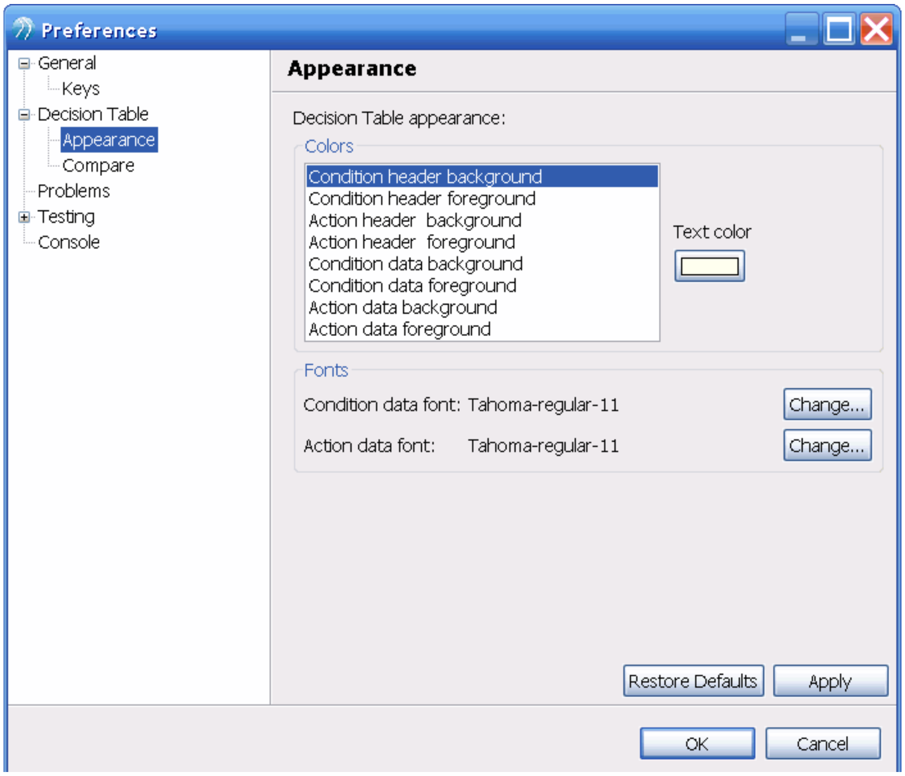
On [✓ Off]

Advanced

- [ ] Use Non-resizable Editor Sections
- [ ] Display column ID
- [✓] Use existing IDs when importing

You can set Table View, Editor menu options, Condition field filter, Action field filter, and Advanced preferences for decision tables. You can also set preferences for different cells of a decision table with colors and fonts.

Decision Table—Appearance Preferences



Decision Table appearance:

Colors

- Condition header background
- Condition header foreground
- Action header background
- Action header foreground
- Condition data background
- Condition data foreground
- Action data background
- Action data foreground

Text color

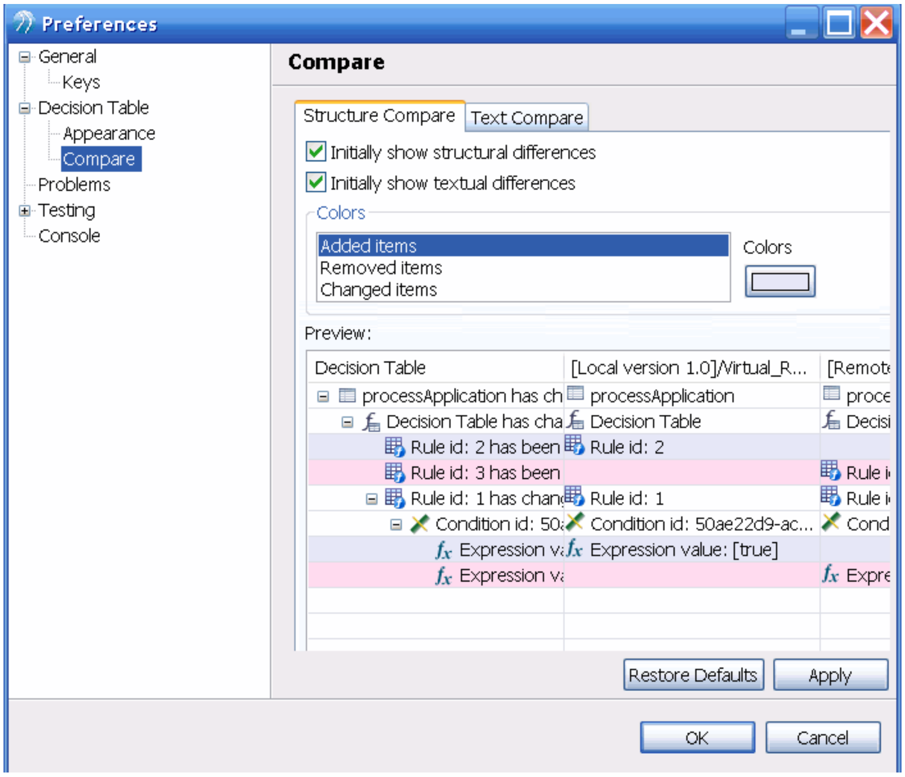
Fonts

Condition data font: Tahoma-regular-11 Change...

Action data font: Tahoma-regular-11 Change...

For each item in the list, you can set color preferences. For condition data and action data you can specify a font.

Decision Table—Compare Preferences



Preferences

- General
  - Keys
- Decision Table
  - Appearance
  - Compare
- Problems
- Testing
- Console

Compare

Structure Compare | Text Compare

- [✓] Initially show structural differences
- [✓] Initially show textual differences

Colors

- Added items
- Removed items
- Changed items

Colors

Preview:

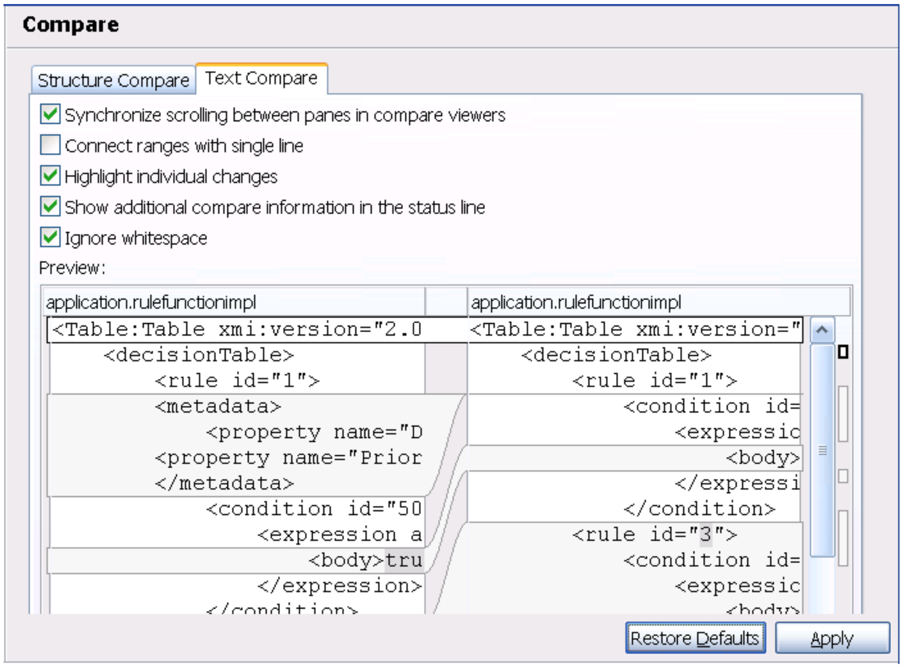
Decision Table	[Local version 1.0]/Virtual_R...	[Remote
processApplication has ch	processApplication	proce
Decision Table has cha	Decision Table	Decisi
Rule id: 2 has been	Rule id: 2	
Rule id: 3 has been		Rule i
Rule id: 1 has chan	Rule id: 1	Rule i
Condition id: 50a	Condition id: 50ae22d9-ac...	Cond
Expression va	Expression value: [true]	
Expression va		Expre

Restore Defaults | Apply

OK | Cancel

In the Structure Compare tab (shown above), set preferences for initial settings and colors. the effect of your choices is shown in the lower panel.

In the Text Compare tab (shown below), set preferences for text. The effect of your choices is shown in the lower panel.



- [✓] Synchronize scrolling between panes in compare viewers
- [ ] Connect ranges with single line
- [✓] Highlight individual changes
- [✓] Show additional compare information in the status line
- [✓] Ignore whitespace

Preview:

```
application.rulefunctionimpl          application.rulefunctionimpl
<Table:Table xmi:version="2.0         <Table:Table xmi:version="
    <decisionTable>                       <decisionTable>
        <rule id="1">                         <rule id="1">
        <metadata>                                <condition id=
            <property name="D                         <expressic
        <property name="Prior                             <body>
        </metadata>                                   </expressi
            <condition id="50                     </condition>
                <expression a                 <rule id="3">
                    <body>tru                     <condition id=
                </expression>                         <expressic
            </condition>                                  <body>
```

Problems Preferences

When you validate a decision project, errors such as language validations and access control validations may occur, and they are shown in Problems view. You can set display and sorting preferences for the Problems view window. These preferences are described in Table 10.

Table 10 Problems view preferences

Label	Notes
Problems View	
`Maximum number of items to show`	Allows you to set the limit the number of problems displayed in the view.
`Sort By`	You can select the filter from the drop down menu to sort the problems.
`Show Resource column`	If checked, the Resource column is displayed.

Table 10 Problems view preferences

Label	Notes
`Activate Problems view when new Problems are found`	If checked, the Problems view is displayed with complete problem description whenever a problem occurs.

Testing Preferences

One option is available: Automatically deploy all tables before starting project tests.

If you do not automatically deploy all tables before starting project tests, then you must use the menu options to deploy tables for testing. You can either select and deploy individual tables for testing, or deploy all tables for testing.

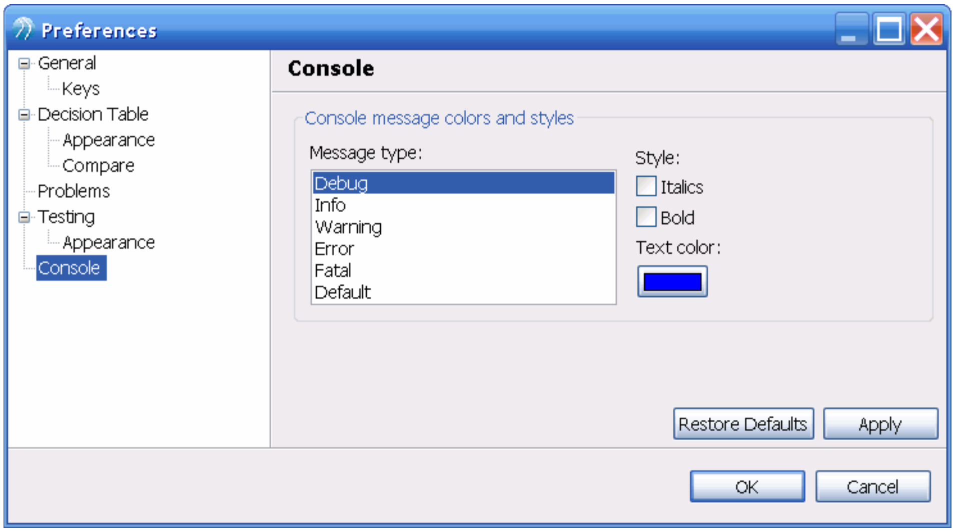
If you enable the option to automatically deploy all tables before starting project tests, then you can't specify which tables to test. They are all deployed for testing when you start the test. However the procedure is simpler because you don't have to explicitly deploy the tables.

Testing Preferences—Appearance

When you test a decision project, Decision Manager tester shows the values changed while running the BE engine and the new instances created in Result Viewer. You can set the text color preferences for: Changed value background, Changed value foreground, New Instances background, and New Instances foreground.

Console Preferences

You can set preferences for each message type, for example using text style and color.



Console

Console message colors and styles

Message type:
- Debug
- Info
- Warning
- Error
- Fatal
- Default

Style:
- [ ] Italics
- [ ] Bold

Text color: 🟦

Index of Engine Properties

Index