

# Using Apama with Software AG Designer

Version 9.10

August 2016

This document applies to Apama Version 9.10 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2013-2016 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

# Table of Contents

<b>About this Guide.....</b>	<b>9</b>
Documentation roadmap.....	9
Online Information.....	11
Contacting customer support.....	12
<b>Overview of Developing Apama Applications.....</b>	<b>13</b>
Starting Software AG Designer.....	14
Demos and tutorials.....	14
The Apama interface.....	15
The Apama Workbench perspective.....	16
Workbench Project view.....	16
The Apama Developer perspective.....	16
Project Explorer view.....	17
The Apama Runtime perspective.....	17
About Apama projects.....	18
Managing project hierarchies.....	19
Editors.....	19
Outline view.....	20
Scenario Browser view.....	20
Engine Receive view.....	21
Engine Status view.....	21
Engine Information view.....	21
Console view.....	22
Problems view.....	23
Data Player Control view.....	23
Building Apama projects.....	23
Launching Apama projects.....	23
<b>Working with Projects.....</b>	<b>25</b>
Creating Apama projects.....	26
Adding resources to Apama projects.....	27
Creating new monitor files for EPL applications.....	27
Creating new event definition files for EPL applications.....	28
Creating event definitions by adding EPL code.....	28
Creating event definitions from XML files.....	29
Creating event definitions from XSD files.....	30
Creating new files for JMon applications.....	31
Adding a new JMon application.....	31
Adding a JMon monitor.....	31
Adding a JMon event.....	32
Adding an EPL Plug-in written in Java.....	33

Creating new scenarios.....	34
Creating new blocks.....	35
Creating a block with the block editor.....	35
Creating a block from an EPL event definition.....	36
Adding EPL code to a block.....	38
Creating new scenario functions.....	38
Creating new dashboards.....	39
Creating new dashboard-deployment configurations.....	40
Creating new event files.....	41
Adding resources to EPL projects.....	42
Adding resources to JMon projects.....	42
Adding JMon applications.....	43
Adding JMon classes.....	43
Adding non-JMon Java files.....	43
Adding bundles to projects.....	44
Bundle instances.....	44
Adding adapters to projects.....	45
Adding Universal Messaging configuration to projects.....	46
Editing Apama files.....	46
Obtaining content assistance.....	47
Using auto-completion.....	47
Displaying information for events and actions.....	47
Specifying comments.....	47
Using auto-indent.....	48
Using auto-bracketing.....	48
Using tabs.....	48
Defining shorthand (templates) for frequently used EPL code.....	48
Sharing templates among Software AG Designer installations.....	49
Specifying colors to distinguish EPL elements.....	49
Shortcuts when editing Apama files.....	49
Navigating in Apama files.....	51
Using the Outline view to navigate.....	51
Using the Quick Outline to navigate.....	51
Jumping to an event or action definition or variable declaration.....	51
Searching in EPL files.....	51
Building Apama projects.....	52
Build automatically when a resource changes.....	53
Build all Apama projects.....	53
Build one Apama project.....	53
Build a working set.....	53
Clean and rebuild projects.....	54
Configuring the project build path.....	54
Project source files.....	54
Specifying projects.....	55
Specifying external dependencies.....	55

Specifying dependencies for a single-user project.....	55
Specifying dependencies for a multi-user project.....	56
Defining MonitorScript Build Path variables.....	56
Importing projects.....	57
Importing adapter configurations.....	57
Setting up the environment before importing projects.....	58
Exporting project information.....	59
Exporting a project initialization file list.....	60
Exporting to a deployment script.....	60
Exporting scenarios.....	62
Exporting Correlator Deployment Packages.....	62
Exporting adapter configurations.....	63
Exporting ApamaDoc.....	64
Deleting projects and resources.....	64
Deleting resources.....	64
Deleting projects.....	64
Adding the Apama nature to a project.....	65
Adding the Java nature to an Apama project.....	65
Viewing all EPL objects in all projects.....	66
Internationalizing Apama applications.....	67
Checking the error log.....	67
Using Software AG Designer to configure adapters that use UM.....	68
<b>Using Query Designer.....</b>	<b>71</b>
Use cases for queries.....	72
Adding query files to projects.....	73
Steps for using Software AG Designer to implement queries.....	74
Overview of creating a query in the Design tab.....	74
Creating queries in Query Designer.....	75
Configuring a query.....	76
Adding query inputs.....	77
Adding query patterns.....	80
Adding query parameters.....	81
Adding query conditions.....	82
Adding query condition filters (where).....	83
Adding query time constraints (within).....	83
Adding query exclusions (without).....	84
Adding query aggregates.....	86
Adding query aggregate calculations.....	86
Adding query aggregate filters.....	87
Adding query actions.....	88
Adding query send event actions.....	89
Adding query custom EPL actions.....	91
Specifying a time period in Query Designer.....	92
Editing source code in Query Designer.....	93

Errors in query definitions.....	94
Enabling diagnostic logging.....	95
Configuring query projects.....	95
Exporting query deployment scripts.....	95
<b>Launching Projects.....</b>	<b>97</b>
Running Apama projects.....	98
Default launch configuration.....	98
Workbench perspective.....	99
Developer perspective.....	99
Defining custom launch configurations.....	99
Adding a correlator.....	102
Correlator arguments.....	102
Persistence options.....	103
Injections.....	104
Event Files.....	105
Connecting correlators.....	105
Adding an external process.....	106
Testing a subset of your apama application.....	106
Monitoring Apama applications.....	107
Console view.....	107
Using the Engine Information view.....	108
Using the Engine Receive view.....	109
Engine Receive Viewer preferences.....	109
Using the Engine Status view.....	109
Using the Scenario Browser view.....	111
Displaying the Scenario Browser.....	111
Browsing scenarios and query definitions.....	111
Creating a new instance of a scenario or a parameterized query.....	112
Viewing the instances of a scenario or a query.....	112
Editing an instance of a scenario or a parameterized query.....	112
Deleting an instance of a scenario or a parameterized query.....	113
Deleting all instances of a scenario or a parameterized query.....	113
Dashboards.....	113
<b>Debugging EPL Applications.....</b>	<b>115</b>
Adding breakpoints.....	116
Launching a debug session.....	116
Creating a debug configuration.....	117
Debugging a remote application.....	117
Debug view for an EPL application.....	118
Breakpoints view for an EPL application.....	118
Variables view for an EPL application.....	119
Command-line debugger.....	119
<b>Debugging JMon Applications.....</b>	<b>121</b>

Preparing the correlator for remote debugging.....	122
Creating a debug run configuration.....	123
Debug perspective.....	124
Using the Debug view with a JMon application.....	124
Working with breakpoints in a JMon application.....	124
Viewing stack frame variables for a JMon application.....	125
Example debug session.....	125
Debug example of preparing code and JAR file.....	125
Debug example of starting correlator and injecting application.....	126
Example of debugging in Software AG Designer.....	127
Additional resources for Java debugging.....	127
<b>Profiling EPL Applications.....</b>	<b>129</b>
Launching profiling sessions.....	130
Launching a default profiling session.....	130
Launching a custom profiling session.....	131
Creating a custom profile launch configuration.....	131
Launching a remote profiling session.....	132
Creating a remote profiler launch configuration.....	132
The Apama Profiler perspective.....	133
Profiling Monitor view.....	133
Execution Statistics view.....	133
The Execution Statistics tab.....	134
Comparison of Execution Statistics tab.....	134
Viewing EPL code.....	135
Using filters.....	135
Creating a Filter.....	135
Managing Filters.....	136
Taking snapshots.....	136
Using snapshots.....	137
Choosing profiling information columns.....	138
Updating profile data.....	138
Displaying Apama perspective preferences.....	138
<b>Using the Data Player.....</b>	<b>141</b>
Introduction to the Data Player.....	142
Using the Data Player.....	142
Adding the ADBC adapter.....	142
Launching the project.....	143
Specifying Data Player playback queries.....	144
Data Player Control view.....	147
Playback settings.....	147
Playback controls.....	147
Playback status.....	147
Creating Data Player query templates.....	148
Command-line Data Player interface.....	149

---

<b>Setting the Apama Preferences.....</b>	<b>151</b>
Showing the Apama-specific preferences.....	152
Apama.....	152
Adapters.....	153
Web Service.....	153
Catalogs.....	154
MonitorScript.....	154
Editor Colors.....	155
Editor Formatting.....	155
Editor Templates.....	156
MonitorScript Path Variables.....	157
Profiling and Logging.....	158
Query.....	159
Syntax Coloring.....	159
Scenarios.....	160
Workbench.....	161



## About this Guide

---

This guide explains how to develop Apama applications in Software AG Designer, which is an Eclipse-based integrated development environment.

## Documentation roadmap

---

Apama provides documentation in the following formats:

- HTML (viewable in a web browser)
- PDF (available from the documentation website)
- Eclipse help (accessible from the Software AG Designer)

You can access the HTML documentation on your machine after Apama has been installed:

- **Windows.** Select **Start > All Programs > Software AG > Tools > Apama *n.n* > Apama Documentation *n.n***. Note that **Software AG** is the default group name that can be changed during the installation.
- **UNIX.** Display the `index.html` file, which is in the `doc` directory of your Apama installation directory.

The following table describes the different guides that are available.

Title	Description
<i>Release Notes</i>	Describes new features and changes since the previous release.
<i>Installing Apama</i>	Instructions for installing Apama.
<i>Introduction to Apama</i>	Introduction to developing Apama applications, discussions of Apama architecture and concepts, and pointers to sources of information outside the documentation set.
<i>Using Apama with Software AG Designer</i>	Instructions for using Apama to create and test Apama projects, develop EPL programs, define Apama queries, develop JMon programs, and store, retrieve and play back data.
<i>Developing Apama Applications</i>	Describes the different technologies for developing applications: EPL monitors, Apama queries, Event Modeler, and Java. You can use one or several of

Title	Description
	<p>these technologies to implement a single Apama application. In addition, there are C++, C, and Java APIs for developing components that plug in to a correlator. You can use these components from EPL.</p>
<p><i>Connecting Apama Applications to External Components</i></p>	<p>Describes how to connect Apama applications to any event data source, database, messaging infrastructure, or application. The general alternatives for doing this are as follows:</p> <ul style="list-style-type: none"> <li>■ Implement standard Apama Integration Adapter Framework (IAF) adapters.</li> <li>■ Create applications that use correlator-integrated messaging for JMS or Software AG's Universal Messaging.</li> <li>■ Use connectivity plug-ins written in Java or C++.</li> <li>■ Develop adapters with Apama APIs for Java and C++.</li> <li>■ Develop client applications with Apama APIs for Java, .NET, and C++.</li> </ul>
<p><i>Building and Using Dashboards</i></p>	<p>Describes how to build and use an Apama dashboard, which provides the ability to view and interact with scenarios and DataViews. An Apama project typically uses one or more dashboards, which are created in the Dashboard Builder. The Dashboard Viewer provides the ability to use dashboards created in Dashboard Builder. Dashboards can also be deployed as simple Web pages, applets, or WebStart applications. Deployed dashboards connect to one or more correlators by means of a Dashboard Data Server or Display Server.</p>
<p><i>Deploying and Managing Apama Applications</i></p>	<p>Describes how to deploy components with Command Central or with Apama's Enterprise Management and Monitoring (EMM) console. Also provides information for:</p> <ul style="list-style-type: none"> <li>■ Improving Apama application performance by using multiple correlators and saving and reusing a snapshot of a correlator's state.</li> <li>■ Managing and monitoring over REST (Representational State Transfer).</li> </ul>

Title	Description
	■ Using correlator utilities.

In addition to the above guides, Apama also provides the following API reference information:

- API Reference for EPL in ApamaDoc format
- API Reference for Java in Javadoc format
- API Reference for C++ in Doxygen format
- API Reference for .NET in HTML format

## Online Information

---

### Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

### Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

### Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

## Contacting customer support

---

If you have an account, you may open Apama Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>. If you do not yet have an account, send an email to [empower@softwareag.com](mailto:empower@softwareag.com) with your name, company, and company email address and request an account.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at [https://empower.softwareag.com/public\\_directory.asp](https://empower.softwareag.com/public_directory.asp) and give us a call.

# 1 Overview of Developing Apama Applications

---

■ Starting Software AG Designer .....	14
■ Demos and tutorials .....	14
■ The Apama interface .....	15
■ The Apama Workbench perspective .....	16
■ The Apama Developer perspective .....	16
■ The Apama Runtime perspective .....	17
■ About Apama projects .....	18
■ Building Apama projects .....	23
■ Launching Apama projects .....	23

Apama runs in Software AG Designer, which provides an integrated environment for developing Apama applications. The process of developing an Apama application is centered around an Apama project. In Software AG Designer, you create an Apama project and then proceed as follows:

- Create new Apama resources for the application.
- Include standard, pre-packaged Apama resources.
- Include existing Apama resources from other projects or applications.
- Specify configuration properties necessary for launching the application.
- Run and monitor the application.
- Export the initialization information necessary for deploying the application.

When you add resources to your application, Software AG Designer creates the resource's metadata and launches the appropriate editor where you add the code to implement its behavior. As you create the resource's application code, Software AG Designer automatically validates it. Where necessary, Software AG Designer launches the Apama tool (such as Query Designer, Event Modeler or Dashboard Builder) that is appropriate to the specific resource being added.

## Starting Software AG Designer

---

When you start Software AG Designer for the first time, you are prompted to select the workspace that is to be used. All projects that you will create will be stored in this workspace folder. By default, this is `C:\Users\username\workspace\...`.

### To start Software AG Designer

- Choose the following from the Windows Start menu: **All Programs > Software AG > Tools > Software AG Designer *n.n***. Note that **Software AG** is the default group name that can be changed during the installation.

## Demos and tutorials

---

Apama provides several demo applications and tutorials in Software AG Designer. These are available from the Welcome page, which appears when you start Software AG Designer for the first time and which is always available by choosing **Help > Welcome** from the Software AG Designer menu. On the Welcome page, look for the heading **Apama**, and then click **Demos** or **Tutorials**.

The Apama demos are demonstration applications that illustrate some of the features and capabilities of the Apama platform. They include dashboards in which you can interact with the application and readme files that describe what the application does, what files make up the project, and how you might modify the application. Use the

demonstration applications to gain an overview of what goes into an Apama project and what some common Apama applications can do.

The tutorials are interactive instructions that get you quickly up to speed writing Event Processing Language programs, creating scenarios, defining reusable blocks for use in scenarios, and creating dashboards that provide the user interface to scenarios. Each tutorial provides a skeleton project and a completed project. At the end of the tutorial instructions, you run the project.

When you open a demo or a tutorial for the first time, it is copied into the Eclipse workspace. You can revert to the original demo or tutorial without any changes you've made at any time by deleting the project as follows:

1. Right-click the project.
2. Click **Delete**.
3. Select **Also delete contents** in the confirmation dialog, and click **Yes**.

Then open the project again from the **Apama Demos** or **Apama Tutorial** page.

## The Apama interface

---

Software AG Designer provides the following distinct perspectives for working with Apama projects:

- ["The Apama Workbench perspective" on page 16](#)
- ["The Apama Developer perspective" on page 16](#)
- ["The Apama Runtime perspective" on page 17](#)

In each of the perspectives, you can create projects, add Apama resources, and launch your application. While developing your application, you can switch from one perspective to the other.

When you debug an Apama application, the Eclipse Debug perspective is used by default.

When profiling an Apama application written in the Apama Event Processing Language (EPL), the Apama Profiler perspective is used. See ["The Apama Profiler perspective" on page 133](#).

**Note:** When using any of the Apama perspectives, you can redisplay the default perspective layout by selecting **Window > Perspective > Reset Perspective ...** from the menu.

**Caution:** Care must be taken if Apama was installed in `Program Files`, which is a protected location on recent Microsoft Windows operating systems. These include the client operating systems Windows 7 and Windows 8.1, and the server operating systems Windows Server 2008 R2 and Windows 2012 R2. To write to the `Program Files` folder, you must run the installer with Administrative privileges. After Apama installation, if you want to add

additional plug-ins to Eclipse, you can run the Eclipse plug-in installer or use the Eclipse **Check for Updates** facility but you must have Administrative privileges when you install the Eclipse plug-in. Lack of Administrative privileges might cause the plug-in installation to fail or become corrupt. Administrative privileges are required because Eclipse also installs its plug-ins in the protected `Program Files` folder. Alternatively, you can choose to install Apama in a non-recommended location outside the `Program Files` folder.

## The Apama Workbench perspective

---


The Workbench perspective presents a more streamlined view of a single Apama project. It shows only the resources related to that project and provides a simplified way of launching an Apama application.


The important interface components of the Apama Workbench perspective are:

- ["Workbench Project view" on page 16](#)
- ["Editors" on page 19](#)
- ["Scenario Browser view" on page 20](#)
- ["Console view" on page 22](#)
- ["Problems view" on page 23](#)

### Workbench Project view

When you develop an application in the Apama Workbench perspective, you work on a single project at a time. The project appears in the Workbench Project view. This is where you add the various Apama resources that are necessary for the application. This is also where you run the application that is represented by the project.

When viewing a project in the Workbench perspective, you can use the **Show All Folders** icon  to toggle between displaying a limited view of the project's resources and displaying all of the project's resources.

Clicking the **Show All Folders** icon  displays an expanded view that lists the Event Processing Language (EPL) files and other file types, such as Bundles, block files, and function files used in the project.

## The Apama Developer perspective

---

The Apama Developer perspective uses the full Eclipse interface and displays all the Apama projects in the user's workspace. It is designed for experienced developers and assumes that you are familiar with standard Eclipse features.



The important interface components of the Developer perspective are:

- ["Project Explorer view" on page 17](#)
- ["Console view" on page 22](#)
- ["Outline view" on page 20](#)
- ["Problems view" on page 23](#)
- ["Editors" on page 19](#)

## Project Explorer view

The Apama Developer perspective uses the standard Eclipse Project Explorer view. This view is at the center of the process of developing an application in Software AG Designer. This is where you add and manage all the Apama resources that are necessary for the application. The Project Explorer view is where you build and launch your projects.

## The Apama Runtime perspective

---

The Apama Runtime perspective is similar to the Apama Developer perspective but is designed for inspecting and interacting with a running Apama application. This perspective is designed for experienced developers and assumes that you are familiar with standard Eclipse features.

---

### To open the Apama Runtime perspective

1. Select **Window > Perspective > Open Perspective > Other ...** from the menu.
2. Select **Apama Runtime** from the Open Perspective dialog.
3. Click **OK**.

The Apama Runtime perspective is made up of these views:

- ["Project Explorer view" on page 17](#)
- ["Scenario Browser view" on page 20](#)
- ["Engine Receive view" on page 21](#)
- ["Engine Status view" on page 21](#)
- ["Engine Information view" on page 21](#)
- ["Outline view" on page 20](#)
- ["Console view" on page 22](#)
- ["Problems view" on page 23](#)
- ["Data Player Control view" on page 23](#)

## About Apama projects

---

An Apama project typically manages a single Apama application. A project provides a means of keeping the application's resources organized. In the process of developing an application with Software AG Designer, you add the various resources that make up the application to the project. For example, you can include:













- EPL files — These files define *monitors* and associated *event types* that are used by your application. EPL files have a `.mon` extension.
- Query files — These files define Apama *queries*. Queries process only the event types that you specify. Query files have a `.qry` extension.
- JMon Monitor files — These Java source files define *monitors* for Apama applications written in Java.
- JMon Event files — These Java source files define *event types* for Apama applications written in Java.
- Scenario definition files — If your application uses scenarios, you specify the new scenario in Software AG Designer. This adds a *scenario definition file* to your project and opens the new scenario in Apama's Event Modeler editor where you specify its behavior. Scenario definition files have a `.sdf` extension.
- Dashboard definition files — If your application uses dashboards, you specify the new dashboard in Software AG Designer. This adds a *dashboard definition file* to your project and opens the new dashboard in Apama's Dashboard Builder. Dashboard definition files have an `.rtv` extension.
- Bundles — These are pre-packaged collections of Apama objects.
- Event files — Event files have an `.evt` extension.
- Block definition files — If your application uses scenarios and, in addition to the standard blocks packaged with Apama, you need to create customized blocks, you create the block in Software AG Designer. This adds a *block definition file* to your project. Block definition files have a `.bdf` extension.
- Function definition files — If your application uses scenarios and, in addition to the standard functions packaged with Apama, you need to create customized functions, you create the function in Software AG Designer. This adds a *function definition file* to your project. Function definition files have a `.fdf` extension.


Software AG Designer provides a variety of ways to manage and interact with the resources in your application's project.

This section briefly describes these Apama features. For more information on how to use them, see ["Working with Projects" on page 25](#) and ["Launching Projects" on page 97](#).

## Managing project hierarchies

Software AG Designer organizes your project into a hierarchy that is displayed in the Project Explorer view (if you are using the Developer perspective) or the Workbench Project view (if you are using the Workbench perspective). The hierarchy is made up of folders that group Apama resource files by type. The Project Explorer view lists all your Apama projects, while the **Workbench Project** view lists just the current project. A different icon is shown for each type of resource file:

-  indicates an Apama project.
-  indicates a bundle.
-  indicates an EPL (.mon) file.
-  indicates an Apama project.
-  indicates an Apama query (.qry) file.
-  indicates an EPL plug-in written in Java (.java) file.
-  indicates an event (.evt) file.
-  indicates a scenario (.sdf) file.
-  indicates a scenario block (.bdf) file.
-  indicates a scenario function (.fdf) file.
-  indicates a dashboard (.rtv) file.
-  indicates a correlator deployment package (.cdp) file.

If a monitor, query or scenario file contains an error, Software AG Designer displays an error icon over the file name, over the folder that contains the file that has the error, and over the project folder. For example, a scenario file that contains an error would look like this:  count-days-example.sdf .

## Editors

Apama provides wizards and editors in Software AG Designer to create and modify the resource files that define the following:

- Monitors
- Queries
- Events
- Blocks
- Functions
- Scenarios

## ■ Dashboards

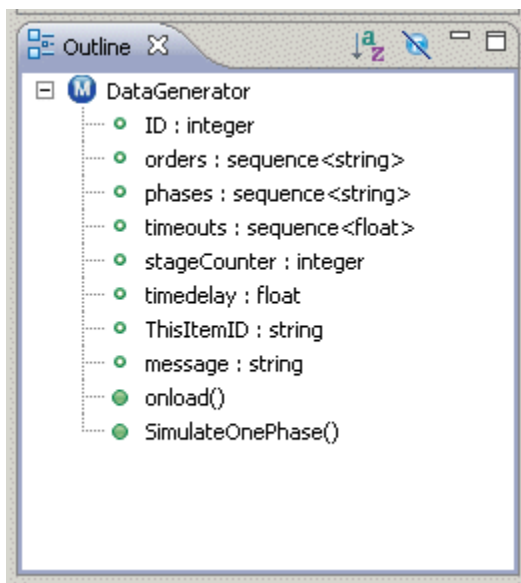
When you add a resource, Software AG Designer generates the definition file and opens it in the appropriate Apama editor.

**Note:** When you add or edit a dashboard, the dashboard's definition file is created and the Apama Dashboard Builder is opened in a separate window.

As you edit resource files, you can take advantage of the Software AG Designer editing features, including content assistance, auto-bracketing, templates for frequently entered constructs, and problem detection. After you build an Apama project, Software AG Designer flags each line that contains an error.

## Outline view

The Outline view shows the structure of the EPL or block file that is open in an Apama editor. The following illustration displays an Outline view of an EPL file.



When you click on an event type or action in the outline, the editor pane highlights and displays the line of code that defines that item in the file.


## Scenario Browser view

The Scenario Browser view displays scenario and query definitions that are loaded in the correlator and any scenario and query instances that are running. When an application is running in Software AG Designer, you can add, edit, and delete scenario and query instances.

For more information on using the view, see ["Using the Scenario Browser view" on page 111](#).

## Engine Receive view

The **Engine Receive** view shows all events generated from the connected correlator.

The **Toggle Connection** button  to temporarily disconnect the **EngineReceive** view from the correlator.

When the **Toggle Connection** button is pressed, the console will not update any further events from correlator. The background of the console will be changed to indicate the temporary disconnect mode. The **Select All**, **Clear**, and **Copy** actions will still work in this mode.

To resume from the temporary disconnect mode, simply click on the **Toggle Connection** button again to resume the connection. The console will be cleared and new events will be shown in the console again.

For more information on the **Engine Receive** view, see ["Using the Engine Receive view" on page 109](#).

## Engine Status view

The **Engine Status** view displays the information about the correlator status. The information is the same as the output of Apama command line tool `engine_watch`.

For more information on the **Engine Status** view, see ["Using the Engine Status view" on page 109](#).

## Engine Information view

The Engine Information view inspects a running correlator and displays defined contents of the correlator. It shows the same information as the Apama command line tool `engine_inspect`. For example:

Aggregates	Type	Monitor Instances	Queue Size	Channels
Contexts	Type			
DV_London	Context	1	0	"London", "location"
DV_New York	Context	1	0	"New York", "location"
DV_Sydney	Context	1	0	"Sydney", "location"
DV_Tokyo	Context	1	0	"Tokyo", "location"
main	Context	7	0	
Events	Type	Templates		
Java Monitors	Type	Listeners		
Monitors	Type	Monitor Instances		
Plugin Receivers	Type		Queue Size	Channels
Receivers	Type	Address	Queue Size	Channels
Timers	Type	Timers		

The following actions are available:

- **Delete** button — Direct deletion of the defined named entities. You can also delete the entity by right clicking the entity and selecting **Delete** from the drop down menu. A confirmation message appears after you click **Delete**. If the entity has dependencies on it, a confirmation message appears asking if you want allow a forced deletion.
- **Send** button — Send an event from this view. You can also send an event by right clicking it and selecting **Send** from the drop down menu.

For more information on the Engine Information view, see ["Using the Engine Information view" on page 108](#).

## Console view

The Console view displays information concerning a running Apama application. An application can have several consoles:

- **Correlator** — Displays output from the correlator.
- **Engine Inject** — Displays initialization information injected to the correlator.
- **Engine Send** — Displays information from Apama components such as dashboards that stream data to the correlator.
- **Correlator Initialization** — Displays information about the correlator initialization including the Java files, `.mon` files (monitors), and `.sdf` files (scenarios) that have been injected and the `.evt` files (events) that have been sent and whether the actions succeeded or failed.

To view one of these consoles, click the drop-down arrow of the **Display Selected Console** button and select the console you want.

## Problems view

The Problems view is a standard Eclipse Problems view that lets you view a list of any errors in your EPL, query and scenario files, and ODBC-ADBC and JDBC-ABDC adapter instances. When you build an Apama project, if Software AG Designer detects errors in your project's files, or file dependency errors, it lists them here. The Problems view appears when you run an application.

## Data Player Control view

The Data Player Control view allows you to control how Apama data is played back in an application running in a project. For example, you can specify how fast to play back the event data, step through events one at a time, or specify a Data Player query to filter what events are played back.

For more information on the Data Player Control, see ["Using the Data Player" on page 141](#).

## Building Apama projects

---

Software AG Designer validates Apama projects, in a process known as “building” the project. During validation, Software AG Designer

- Checks the syntax to ensure that it is valid EPL or Java code.
- Checks that only valid values have been specified. For example, if you specify integer as the type of a field, it ensures that you specify an integer as the value of the field.
- Ensures that the dependencies are valid throughout the project.

In summary, Software AG Designer validates that its launcher can inject the project's EPL files into the correlator.

For more information, see ["Building Apama projects" on page 52](#).

## Launching Apama projects

---

Software AG Designer can launch an Apama project in a test environment. When Software AG Designer launches a project, the default is that it starts an instance of the correlator and injects all the resources that are included in the project. You can change the default launch behavior by editing the launch configuration for the project.

You can specify multiple launch configurations for each Apama project, including configurations for debugging and profiling applications. For each launch configuration, you can:

- Add correlator arguments.
- Specify event files that should be sent to initialize/start the application.
- Specify what adapters to use.
- Define environment variables, for example, appending directories to the `PATH` used to start the correlator and any IAF processes.
- Indicate whether the configuration can be shared among Apama installations.

For more information on creating launch configurations and launching Apama projects, see ["Launching Projects" on page 97](#).



## 2 Working with Projects

■ Creating Apama projects .....	26
■ Adding resources to Apama projects .....	27
■ Editing Apama files .....	46
■ Navigating in Apama files .....	51
■ Building Apama projects .....	52
■ Importing projects .....	57
■ Importing adapter configurations .....	57
■ Setting up the environment before importing projects .....	58
■ Exporting project information .....	59
■ Deleting projects and resources .....	64
■ Adding the Apama nature to a project .....	65
■ Adding the Java nature to an Apama project .....	65
■ Viewing all EPL objects in all projects .....	66
■ Internationalizing Apama applications .....	67
■ Checking the error log .....	67
■ Using Software AG Designer to configure adapters that use UM .....	68

This section describes how to use Software AG Designer to develop Apama applications.

You can use Software AG Designer to write Apama applications in the Apama Event Processing Language (EPL) and in the Apama in-process API for Java (JMon).

As you develop your application to a stage where you want to run it for testing purposes, you can launch it directly from Software AG Designer. For more information on launching Apama projects, see ["Launching Projects" on page 97](#).

## Creating Apama projects

---

### To create an Apama project

1. From the Software AG Designer menu, select **File > New > Apama Project** to display the New Project dialog. If you are using the Apama Workbench perspective, you can also click the **New Project** button in top right of the Workbench Project view.
2. In the New Apama Project dialog, specify information for the following fields:
  - a. In the **Project Name** field, enter the name of your new project. Project names typically use title case capitalization and may contain spaces, for example. "My Project".
  - b. If you want to store the project in a directory other than the default location, clear the **Use default location** checkbox, click **Browse**, and navigate to and select the location for storing your new project.
3. In the Configure the new project wizard, add checkmarks to the bundles that are appropriate to the type of application you are developing. For example, if your application is a dataview application, select the DataView Service bundle. When you create a new project, the settings for the selected bundles remain same as they were for the last time you created a new project. You can change these settings and specify other bundles for your application. Click **Next**.

Bundles package service monitors, event definitions and other files associated with the standard Apama adapters that are required for the type of application you are building. For example, applications that include scenarios need the Scenario Service bundle; this is selected by default. Applications that use data views need the DataView bundle. If your application requires an adapter, select the associated bundle from this list.

If you are creating a project for an EPL application, click **Finish**.

If you are creating a project for a JMon application or developing Java connectivity plug-ins, select **Add Apama Java support**. If you want to apply standard Eclipse Java support, select **Add Java support**. See ["Adding the Java nature to an Apama project" on page 65](#) for detailed information on these options. Click **Next**.

4. If you have selected **Add Apama Java support**, the Configure the new Apama Java application dialog appears. Fill in the fields as desired. Click **Finish**.

5. If you are not currently in an Apama perspective, the Open Associated Perspective? dialog appears; it is up to you whether to select **Remember my decision**. Click **Yes** or **No**.

If you are in the Apama Developer perspective, Software AG Designer displays the name of your new project in the Project Explorer view pane on the left of the perspective. If this is your first project, the other panes are blank.

## Adding resources to Apama projects

You specify the Apama resources that make up your application by adding them to the Apama project. Software AG Designer recognizes and provides editing features for the following files:

- EPL files (`.mon`)
- Apama query files (`.qry`)
- Block definition files (`.bdf`)
- Function definition files (`.fdf`)
- Event files (`.evt`)
- JMon monitor and event files (`.java`)
- Scenario definition files (`.sdf`)
- Dashboard definition files (`.rtv`) — opens in Dashboard Builder

However, you can add any type of file to an Apama project and Eclipse uses the correct editor to open it.

When adding new blocks and functions to a project, you should create the definition files using Software AG Designer.

## Creating new monitor files for EPL applications


EPL files define monitors and/or event types or they define Apama queries.

**Note:** EPL monitors and EPL queries (also referred to as Apama queries) provide two different approaches to implementing an Apama application. This topic provides instructions for adding files that will define monitors and/or events. To add a file that will define a query, see ["Adding query files to projects" on page 73](#). See also: "Comparison of queries and monitors" in *Introduction to Apama*.

Monitor definitions and event type definitions are in `.mon` files.

---

### To add a new `.mon` file to an Apama project

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.
2. In the Project Explorer view or the Workbench Project view, right-click the `monitors` folder of the project where you want to add the EPL file and select **New > MonitorScript File**. In the Workbench Project view you can also select the `monitors` folder and click **New MonitorScript File**.
3. In the New MonitorScript File wizard, enter information in the following fields:
  - a. The **Containing Folder** field is the folder where the file will be saved; by default this is the folder of the currently selected project, but you can select another folder using the **Browse** button.
  - b. In the **File name** field, specify the name of the new file. Specifying the `.mon` extension is optional as Software AG Designer will add the `.mon` file extension. Software AG Designer will not let you specify anything except `.mon` as a file extension.
  - c. The **Package** field is optional; information in this field is an EPL package name.
4. Click **Finish**. The name of the new file now appears in the Project Explorer view or the Workbench Project view under the project that contains it and the EPL file opens in the EPL editor.
5. In the EPL editor, add the desired EPL code and save the file.

For more information about EPL, see "Getting Started with Apama EPL" in *Developing Apama Applications*. For more information on the editing features available when writing EPL code, see ["Editing Apama files" on page 46](#).

## Creating new event definition files for EPL applications

You can add new event definitions to an Apama project by adding EPL code. You can also add new event definitions to an Apama project from XML files and XSD files.

### Creating event definitions by adding EPL code

---

#### To create an event definition by adding EPL code

1. In the Project Explorer view, right-click the project's `eventdefinitions` folder and select **New > Event Definition** from the pop-up menu. The New Event Definition dialog appears.
2. In the New Event Definition dialog, in the **Generate Event Definition using** field, select **EPL Editor** and click **Next**. The Choose output location for Event Definition dialog appears.
3. In the Choose output location for Event Definition dialog, enter information in the following fields:

- a. The **Containing Folder** field is the folder where the file will be saved; by default this is the currently selected folder of the current project, but you can select another folder or project using the **Browse** button.
  - b. In the **File name** field, specify the name of the new file. Specifying the `.mon` extension is optional as Software AG Designer will add the `.mon` file extension. Software AG Designer will not let you specify anything except `.mon` as a file extension.
  - c. The **Package** field is optional; information in this field is an EPL package name.
4. Click **Finish**. The new event definition file is added to the specified folder in the project and the EPL file opens in the EPL editor.
  5. In the EPL editor, add the desired EPL code and save the file.

Note, you can also create these types of event definitions by using the **New > MonitorScript** menu item; for details, see ["Creating new monitor files for EPL applications" on page 27](#).

## Creating event definitions from XML files

You can create an event definition that is based on the structure of an XML document. The generated event definition is based on the following:

- All fields are treated as `string`.
- All prefixes are ignored, in other words, event definitions will be generated even if inner elements have different namespaces.
- Namespace attributes (attributes having an `xmlns` prefix) are ignored.
- All inner event names are prefixed with `Element_`.
- Generated event definitions will have an `xmlTextNode` field if the XML element contains a text value and also contains an attribute.

---

### To create an event definition from an XML file

1. In the Project Explorer view, right-click the project's `eventdefinitions` folder and select **New > Event Definition** from the pop-up menu. The New Event Definition dialog appears.
2. In the New Event Definition dialog, in the **Generate Event Definition using** field, select **XML File** and click **Next**. The Choose output location for Event Definition dialog appears.
3. In the Choose output location for Event Definition dialog, enter information in the following fields:
  - a. The **Containing Folder** field is the folder where the file will be saved; by default this is the currently selected folder of the current project, but you can select another folder or project using the **Browse** button.

- b. In the **File name** field, specify the name of the new file. Specifying the `.mon` extension is optional as Software AG Designer will add the `.mon` file extension. Software AG Designer will not let you specify anything except `.mon` as a file extension.
  - c. The **Package** field is optional; information in this field is an EPL package name.
4. Click **Next**. This displays the Select XML File dialog.
5. In the Select XML File dialog, in the **XML File** field, specify the name of the XML file on which you want to base the event. You can use the **Browse** button to navigate to the desired file. The Down Arrow button switches the scope between Workplace and FileSystem.
6. In the Select XML File dialog, in the **Event Name** field, specify the name you want to assign to the root level event.
7. Click **Finish**.

An EPL file is created that defines the root level event along with associated nested events.

## Creating event definitions from XSD files

You can create event definitions that are based on the structures of elements defined in XSD schema files.

---

### To create an event definition from an XSD file

1. In the Project Explorer view, right-click the project's `eventdefinitions` folder and select **New > Event Definition** from the pop-up menu. The New Event Definition dialog appears.
2. In the New Event Definition dialog, in the **Generate Event Definition using** field, select **XSD File** and click **Next**. The Choose output location for Event Definition dialog appears.
3. In the Choose output location for Event Definition dialog, enter information in the following fields:
  - a. The **Containing Folder** field is the folder where the file will be saved; by default this is the currently selected folder of the current project, but you can select another folder or project using the **Browse** button.
  - b. In the **File name** field, specify the name of the new file. Specifying the `.mon` extension is optional as Software AG Designer will add the `.mon` file extension. Software AG Designer will not let you specify anything except `.mon` as a file extension.
  - c. The **Package** field is optional; information in this field is an EPL package name.
4. Click **Next**. This displays the Select XSD File dialog.

5. In the Select XSD File dialog, click the **Browse** button to the right of the **Schema Element/Type** field. This displays the Type Chooser dialog.
6. In the Type Chooser dialog, specify the file that contains schema's global element that you want to use as the root element on which to base the event definition. Click the **Browse** button to navigate to the file. The Drop-down arrow allow you to change scope among Recent files, Local file system, Workspace, Remote URL, and XML Schema. After selecting a file, click **OK**. This returns to the Select XSD File dialog with the root element you selected displayed in the **Schema Element/Type** field.
7. In the Select XSD File dialog, in the **Event Name** field, specify the name you want to assign to the root level event.
8. Click **Finish**.


An EPL file is created that defines the root level event along with associated nested events.

## Creating new files for JMon applications

For Apama projects that use JMon applications, you can add new JMon applications, JMon monitors, and JMon events. In Apama applications written in Java, monitors and event types are implemented as Java classes.


### Adding a new JMon application

#### To add a new JMon application to an Apama project

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.
2. In the Project Explorer view or the Workbench Project view, right-click the name of the project where you want to add the monitor and select **New > Java Application**. The New Java Application wizard appears.
3. In the New Java Application wizard, in the **Configuration** field specify the name of the application and click **Finish**. The application is added to the project and its configuration opens in the Apama Java configuration editor.
4. In the Apama Java configuration editor, specify the application's meta-data and add classes and contents as required. For more information on editing the configuration, see ["Adding resources to JMon projects" on page 42](#).

### Adding a JMon monitor

#### To add a new JMon monitor to an Apama project

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.
2. In the Project Explorer view or the Workbench Project view, right-click the name of the project where you want to add the monitor and select **New > Java Monitor**. You can


also select **File > New > Other** from the menu and then select **Apama > Java Monitor** from the Select a wizard dialog. The New Apama Java Monitor wizard appears.

3. In the New Apama Java Monitor wizard, enter information in the following fields:
  - a. The **Application** field is the application to which you are adding the monitor.
  - b. In the **Monitor name** field, specify the name of the new monitor. This will become the name of the class and the Java file.
  - c. The **Description** field is optional.
  - d. The **Apama Package** field is optional; this is the package of the monitor inside the correlator.
  - e. Add a check to the **Implement MatchListener** check box if you want Software AG Designer to generate the skeleton code for the class's `MatchListener` interface.
  - f. The **Source Folder** field specifies the folder in the project to contain the file; by default this is `java/src`.
  - g. The **Java Package** field is optional; this is the package of the created Java class.
4. Click **Finish**. The name of the new class file now appears in the Project Explorer view or the Workbench Project view under the project that contains it and the `.java` file opens in the editor.

For more information about JMon applications, see "Overview of JMon Applications" in *Developing Apama Applications*.

## Adding a JMon event

### To add a new JMon event to an Apama project

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.
2. In the Project Explorer view or the Workbench Project view, right-click the name of the project where you want to add the monitor and select **New > Java Event > Other**. You can also select **File > New** from the menu and then select **Apama > Java Event** from the Select a wizard dialog. The New Apama Java Event wizard appears.
3. In the New Apama Java Event wizard, enter information in the following fields:
  - a. The **Application** field is the application to which you are adding the event.
  - b. In the **Monitor name** field, specify the name of the new event. This will become the name of the class and the Java file.
  - c. The **Description** field is optional.
  - d. The **Apama Package** field is optional; this is the package of the event inside the correlator.
  - e. The **Source Folder** field specifies the folder in the project to contain the file; by default this is `java/src`.






- f. The **Java Package** field is optional; this is the package of the created Java class.
4. Click **Finish**. The name of the new class file now appears in the Project Explorer view or the Workbench Project view under the project that contains it and the `.java` file opens in the editor.

For more information about JMon applications, see "Overview of JMon Applications" in *Developing Apama Applications*.

## Adding an EPL Plug-in written in Java

### To add a new correlator plug-in written in Java to an Apama project

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.
2. In the Project Explorer view or the Workbench Project view, right-click the name of the project where you want to add the correlator plug-in and select **New > Java EPL Plugin**. You can also select **File > New > Other** from the menu and then select **Apama > Java EPL Plugin** from the Select a wizard dialog. The New Apama Java EPL Plugin wizard appears.
3. In the New Apama Java EPL plug-in wizard, enter information in the following fields:
  - a. The **Application** field is the application to which you are adding the plug-in.
  - b. In the **Plugin name** field, specify the name of the new plug-in. This will become the name of the class and the Java file.
  - c. The **Description** field is optional.
  - d. The **Apama Package** field is optional; this is the package of the plug-in inside the correlator.
  - e. The **Source Folder** field specifies the folder in the project to contain the file; by default this is `java/src`.
  - f. The **Java Package** field is optional; this is the package of the created Java class.
4. Click **Finish**. The name of the new class file now appears in the Project Explorer view or the Workbench Project view under the project that contains it and the `.java` file opens in the editor. In Project Explorer, the  icon indicates a Java plug-in to the correlator.
5. Select the **Show All Folders**  icon in the **Project View** pane. You will find the source file for your Java class in the `java/src` package in your project. You can now add functionality to the class as required for your application. This can be called from EPL using the mechanism described in "Using Java plug-ins" in the "Developing Correlator Plug-ins" part of *Developing Apama Applications*.

For applications that you plan to inject into a correlator, the recommendation is to create separate jar files for:

- EPL plug-ins written in Java
- JMon applications

Although the mechanism for creating these jars and describing their meta-data is similar, the interactions of these two different uses of injected jars mean that they will often need to be injected into the correlator separately. The creation of separate jar files ensures that you can inject your application components in the correct order, which is typically:

1. EPL plug-ins written in Java
2. EPL monitors and events
3. JMon applications

For more information, see "Overview of JMon Applications" as well as "Developing Correlator Plug-ins" in *Developing Apama Applications*.

## Creating new scenarios

Scenarios define templates that, when instantiated with a set of parameters supplied by the user, implement an application's strategy.

---

### To add a new scenario to an Apama project

1. In the Project Explorer view or the Workbench Project view, right-click the `scenarios` folder of the project where you want to add the scenario and select **New > Scenario**. In the Workbench Project view you can also select the `scenarios` folder and click **New Scenario**.
2. In the New Scenario wizard, enter information in the following fields:
  - a. The **Containing Folder** field is the folder where the definition file will be saved; by default this is the folder of the currently selected project, but you can select another folder using the **Browse** button.
  - b. In the **File name** field, specify the name of the new file. Specifying the `.sdf` extension is optional as Software AG Designer will add the `.sdf` file extension. Software AG Designer will not let you specify anything except `.sdf` as a file extension.
3. Click **Finish**. The name of the new scenario definition file is added to project and the new scenario is opened in the Apama Event Modeler.

Using Event Modeler, you complete the scenario by adding state and rules and specify the blocks and functions necessary to implement the desired strategy. For more details on developing scenarios, see "Overview of Using Event Modeler" in *Developing Apama Applications*.

## Creating new blocks

This section is a brief description of how to create a new block in Software AG Designer. For more details on this process, see "Creating Blocks" in the Event Modeler part of *Developing Apama Applications*. You can create a block in Software AG Designer either from scratch or from an existing event definition.

**Note:** When you create a new block, you should place it in the project's default blocks directory. This directory is found in the project's catalogs directory. The blocks directory has a name in the form `project_name blocks`. So, for example, the default block directory of a project named `My_Project` will be `catalogs\My_Project blocks`. If you place the block in the default block directory, scenarios created in the project will automatically find them and make them available in Event Modeler when you are displaying the scenario.

## Creating a block with the block editor

Creating a block in Software AG Designer consists of two main steps. In the first step you create the block metadata and specify its interface. In the second step you add the EPL code that implements the block's behavior.

---

### To add a new block to an Apama project using the block editor

1. In the Project Explorer or Workbench Project view, right-click the default block folder (typically `catalogs\project_name blocks`) and select **New > Block** button from the pop-up context menu. The New Event Modeler Block dialog appears.
2. In the New Event Modeler Block dialog, specify the name of the project in the **Containing** folder field if you want to create the block in a different project; you can also click **Browse** to select the project from a list of projects in the workspace.
3. Also in the New Event Modeler Block dialog, select **Block Editor** in the **Generate block using** field and click **Next**. The Create a new Apama Block dialog appears.
4. In the Create a new Apama Block dialog, fill in the various fields or accept the default values. The most important fields to set at this point are the block name and version. The **Type** field specifies what type of code will be generated for the block. The choices are:
  - **Callback** — Implements the block as an event type; this is the default. Generates EPL code to which you can add custom code.
  - **Callback (DEBUG)** — Implements the block as an event type. Generates EPL code to which you can add custom code. Also generates statements that can help you debug the EPL you are adding. You can easily switch your block between Callback and Callback (DEBUG).
5. Click **Finish**. The block definition file for the new block is added to the project and the block's metadata appears in the **Builder** tab of the block editor.

6. The left side of the **Builder** tab displays the parameters, input feeds, output feeds, and operations that make up the block; a new block will not contain any entries here. To add one of these elements:
  - a. Right-click the element you want to add and select **Add Parameter**, **Add Input Feed**, **Add Output Feed**, or **Add Operation**. The right side of the **Builder** tab displays the item's properties.
  - b. Fill in the values for the properties.
7. For input feeds and output feeds, right-click the element and select **Add Field**.
8. In the **Properties** panel for the field you added in the previous step fill in the values for the properties and field validation specifications. Repeat for each field you want to add to an input or output feed.
9. When you finish specifying the block's metadata, save the file.

For details about adding EPL code to a block, see ["Adding EPL code to a block" on page 38](#).

## Creating a block from an EPL event definition

In cases where you want to create an event from a scenario or to update the scenario types from an Apama event, you should create a block based on the event type.

Creating a block from an existing EPL event definition consists of selecting the event definition from which you want to create the block, specifying whether you want to create an input block or an output block, and specifying what fields in the event are to be used when the block is generated. For input blocks, the specified fields are used to construct listeners; for output blocks, the specified fields are used to set the values of output parameters.

---

### To add a new block to an Apama project from an existing EPL event definition

1. In the Project Explorer or Workbench Project view, right-click the default block folder (typically `catalogs\project_name\blocks`) and select **New > Block** from the pop-up context menu. The New Event Modeler Block dialog appears.
2. In the New Event Modeler Block dialog, specify the name of the project in the **Containing** folder field if you want to use an event definition in a different project; you can also click **Browse** to select the project from a list of projects in the workspace.
3. Also in the New Event Modeler Block dialog, select **EPL event definition** in the **Generate block using field** and click **Next**. The Select the Block type and Event type dialog appears.
4. In the Select the Block type and Event type dialog, click **Browse** to display the Event Type Selection dialog. Note, that events with field types chunk, listener, or stream are not listed because they cannot be emitted, routed, or enqueued.

- a. In the Event Type Selection dialog's **Choose an Event Type** field, enter the name of the event. As you type, event types that match what you enter are shown in the **Matching Events** list.
  - b. In the **Matching Events** list, select the name of the event type you want to use to generate the block. The name of the EPL file that defines the selected event appears in the status area at the bottom of the dialog.
  - c. Click **OK**.
5. In the Select the Block type and EPL type dialog, specify whether you want to generate an input block or an output block. Also, in the **Include comments** field, add a check if you want to include ApamaDoc comments from the event definition in the generated block code and click **Next**. The Create New Apama Block dialog appears.
  6. In the Create a New Apama Block dialog, specify the name, version, type and description of the new block. By default, Software AG Designer uses the name of the event as the basis for the name of the block. The **Type** field specifies what type of code will be generated for the block. The choices are:
    - **Callback** — Implements the block as an event type; this is the default. Generates EPL code to which you can add custom code.
    - **Callback (DEBUG)** — Implements the block as an event type. Generates EPL code to which you can add custom code. Also generates statements that can help you debug the EPL you are adding. You can easily switch your block between Callback and Callback (DEBUG).

When you finish entering information, click **Next**. The Select Block Parameters from Event Fields dialog appears.

7. In the Select Block Parameters from Event Fields dialog, in the **Event Fields** field, specify the fields to be used to create the block.

Note, for input blocks, fields with inner event fields, and fields of `dictionary` or `sequence` types are not displayed. Also, for both input and output blocks, `constant`, `action`, and `context` fields are not displayed.

- When you create a new input block, code is generated for a block parameter and listener for a specified field. If you select more than one field, block parameters and listeners are created for all combinations of all the selected fields. The maximum number of fields you can select is ten.

For input blocks, code is also generated for an output feed that includes all the fields in the specified event, as well as `start` and `stop` actions that activate and deactivate the listeners.

- When you create a new output block, code is generated for parameters for the event fields you specify. This means that values for those fields can be specified when the block is used.

8. Click **Finish**.

The block's `.bdf` file (the block definition file) is created and opened in the block editor. The left side of the **Builder** tab displays the parameters, input feeds, output feeds, and operations that are automatically generated from the information you specified for the block. For details about adding EPL code to a block, see ["Adding EPL code to a block" on page 38](#).

## Adding EPL code to a block

When you save a block, the underlying code that defines the block's interface is generated and saved as a *block definition file* with a `.bdf` extension. To this file, you then add EPL code to implement the necessary behavior.

### To add code to the block

1. If necessary, double-click the block in the Project Explorer to open it in the block editor. In the block editor display the **Source** tab.
2. On the **Source** tab, code appears either with a gray background or a white background. Code with a gray background is maintained by Software AG Designer and is not editable. The sections of code with a white background are the areas where you add your custom EPL code.


For details, see "Adding EPL code to the block definition" in the Event Modeler part of *Developing Apama Applications*. Also, for background information on the elements that make up the actual code of a block's definition file, see "File Definition Formats", also in the Event Modeler part of *Developing Apama Applications*.

## Creating new scenario functions


Many Apama applications are implemented partially or entirely as scenarios. Apama supplies many pre-packaged scenario functions in Software AG Designer that can be used in conditions or actions when defining rules for scenarios. In addition, you can define your own scenario functions within Software AG Designer.

**Note:** When you create a new function, you should place it in the project's default functions directory. This directory is found in the project's `catalogs` directory. The function directory has a name in the form `project_name functions`. So, for example, the default `functions` directory of a project named `My_Project` will be `catalogs\My_Project functions`. If you place the function in the default function directory, scenarios created in the project will automatically find them and they will be available in Event Modeler when you are displaying the scenario.

### To define a new scenario function

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.

2. In the **Project Explorer** view or the Workbench Project view, open the `catalogs` folder of the project where you want to add the EPL file.
3. Right-click the functions folder within the `catalogs` folder. It will have a name such as `MyProject.functions`. Select **New > Scenario Function**.

In the Workbench Project view you can also select the functions folder and click the **New Scenario Function** button  `New Scenario Function...`.

4. In the New Scenario Function wizard, specify information for the following fields:
  - a. The **Containing Folder** field is the folder where the function definition file will be saved; by default this is the functions folder you selected in Step 4, but you can select another folder using the **Browse** button.
  - b. In the **Function Name** field enter a unique name for the function. By convention, this should be in uppercase.
  - c. In the **Display Name** field enter the name that will be displayed in the Apama Event Modeler tool. By default this the same as the **Function Name** but you can change it if you like; it does not need to be unique.
  - d. In the **Return type** field specify the type for the value returned by the function.
  - e. In the **Description** field specify the descriptive text that will accompany the function in the Apama Event Modeler tool.
5. Click Finish. This generates the definition file for the function and opens the file in the Apama editor.
6. The function definition file is an XML file. To implement the behavior of the function you need add the EPL code to the `action` statement in the file's `<code>` element. In the `.fdf` file, this section is labeled:

```
// TODO: put MonitorScript code to implement the function here
```

After you have added a new scenario function to a project, when you look at a scenario in the project with the Apama Event Modeler, the new function will be included in the Event Modeler's function catalog associated with the project.

For more information on using functions in scenarios, see "Using functions in Event Modeler" in *Developing Apama Applications*. For more information on the XML format of the function definition file, see "File Definition Formats" also in *Developing Apama Applications*.

## Creating new dashboards

Dashboards provide the ability to view and interact with scenarios and DataViews. They contain charts and other objects that dynamically visualize the values of scenario variables and DataView fields. Dashboards can also contain control objects for creating, editing, and deleting scenario instances and DataView items. Normally, you add new dashboards after you have substantially developed and tested your application.



You can create dashboards with either the Dashboard Generation wizard or the Dashboard Builder. The wizard allows you to generate simple, default dashboards, customized with your choices regarding basic layout and visualization objects to use. The Builder is a graphical composition tool that gives you fine-grained control over a dashboard's appearance and behavior.

---

### To create a new dashboard

1. In the Project Explorer view, select the Apama project.
2. In the Project Explorer view or the Workbench Project view, select **New > Dashboard** from the **File** menu.
3. In the New Dashboard dialog:
  - Click **Dashboard Generation wizard** to create dashboards using the Dashboard Generation wizard. Enter a name in the **Configuration name** field. The Dashboard Generation wizard uses this as the name of the new configuration that will be used to generate your dashboards. When you use the wizard, you can accept a default configuration or specify a custom configuration.
  - Click **Dashboard Builder** to create dashboards using the Dashboard Builder.
    - i. **Containing Folder** is the folder where the dashboard definition file will be saved; by default this is the **dashboards** folder of the current project, but you can select another folder using the **Browse** button.
    - ii. **File name** specifies the name of the new definition file. Specifying the **.rtv** extension is optional as Software AG Designer will add the **.rtv** file extension. Software AG Designer will not let you specify anything except **.rtv** as a file extension.
4. Click **Finish**.

If you have selected **Dashboard Generation wizard**, the Dashboard Generation wizard appears, and displays the new dashboard-generation configuration. In addition, a new dashboard-generation configuration file (`dashboard_generation.xml`) appears under the current project's **config** folder, if one wasn't already present. For information on using the Dashboard Generation wizard, see "Generating dashboards" in *Building and Using Dashboards*.

If you have selected **Dashboard Builder**, the name of the new dashboard definition file appears under the current project's **dashboards** folder. Additionally, the Apama Dashboard Builder tool is opened in a separate window, showing the new dashboard. Using the Dashboard Builder, you complete the dashboard by adding visualizing and control objects, and connecting them to live correlator data. For more information on completing dashboards, see "Introduction" in *Building and Using Dashboards*.

## Creating new dashboard-deployment configurations

Dashboard-deployment configurations contain the information necessary for the generation of deployment packages for a project's dashboards.



---

### To create a deployment configuration

1. If you are using the Project Explorer view, ensure that a project is selected.
2. In either the Project Explorer view or the Workbench Project view, select **New > Dashboard Deployment** from the **File** menu. (You can also right-click in the navigation pane and select **Dashboard Deployment** from the popup menu. In the Workbench Project view, you can also click the **New** button that is above the navigation pane and select **Dashboard Deployment** from the **Apama** folder, or click the down arrow that is next to the **New** button, and select **Dashboard Deployment** from the popup menu.)
3. In the New Dashboard Deployment Configuration dialog, enter a name in the **Configuration** field. The Dashboard Deployment Configuration Editor uses this as the name of the new configuration.
4. Click **Finish**. The Dashboard Deployment Configuration Editor appears, and displays the new dashboard configuration. In addition, a new dashboard-deployment configuration file (`dashboard_deploy.xml`) appears under the current project's **config** folder, if one wasn't already present.


For information on using the Dashboard Configuration Deployment Editor, see *Using the Deployment Configuration editor* in *Building and Using Dashboards*.

## Creating new event files

Event files are used to supply events of a specific type. For example in applications written in EPL, you should usually send in an event to tell the monitors making up the application to start listening for events once everything has been injected.

---

### To add a new event file to an Apama project

1. If you are in the Apama Workbench perspective, click the **Show All Folders** icon  if necessary to display the enhanced view that shows all the project's resources.
2. In the Project Explorer view or the Workbench Project view, right-click the `events` folder of the project where you want to add the event file and select **New > File > Event**. In the Workbench Project view you can also select the `events` folder and click the **New Event File** button:



3. In the New Event File wizard, enter information in the following fields:
  - a. The **Containing Folder** field is the folder where the file will be saved; by default this is the folder of the currently selected project, but you can select another folder using the **Browse** button.
  - b. In the **File name** field, specify the name of the new file. Specifying the `.evt` extension is optional as Software AG Designer will add the `.evt` file extension. Software AG Designer will not let you specify anything except `.evt` as a file extension.

4. Click **Finish**. The name of the new file now appears in the Project Explorer view or the Workbench Project view under the project that contains it and the event file opens in the Apama event editor.

## Adding resources to EPL projects

To add an existing EPL file that defines a monitor to an Apama project, you add the file as an External Dependency. You can also add EPL files that are contained in an Apama Correlator Deployment Package file.

---

### To add a file to an active project as an external dependency

1. Select **Project > Properties** from the menu.
2. In the Properties dialog, select **MonitorScript Build Path** from the left hand pane and select the **External Dependencies** tab in the right pane.
3. Click **Add External** button and navigate to the files you want to add. You can select EPL (.mon) files or Correlator Deployment Package (.cdp) files.
4. Click **OK** at the Properties dialog.

## Adding resources to JMon projects

To add existing resources to a JMon project, you modify the project's configuration. The configuration information is stored in the project's `config/apama_java.xml` file.

To add resources to a JMon project, double click the project's `config/apama_java.xml` file. The Apama Java configuration editor appears.

The Application drop-down list on the right of the editor's title bar shows the application you are editing. You can select any JMon application currently included in the project and you can add JMon applications from other projects to your current project.

You can use the **Classpath** field to specify any additional third party jars that should be on the runtime classpath of the JMon application or Java plug-in.

For example, `${sys:MY_THIRD_PARTY_DIR_SYS_PROP}/lib/foo.jar;`  
`${env:MY_OTHER_THIRD_PARTY_DIR_ENV_VAR}/bar.jar`


For more information on `classpath` string, see "Specifying classpath in deployment descriptor files", in *Developing Apama Applications*.

You can add other JMon classes (JMon events and JMon monitors) and non-JMon Java files to a project.

## Adding JMon applications

---

### To add a JMon application from another Apama project

1. Click the **Add application** button (  ). The New Application Name dialog appears.
2. Specify a name for the new application and click **OK**. The editor will display the settings for this new application and the name will be added to the editor's drop down list.
3. In the Apama Java configuration editor, expand the **Java Application Classes** heading. This displays the current list of the Java classes included in the project.
4. Click **Add**.

The Select Java Apama Files dialog appears showing the available JMon Application projects.

5. Select the JMon application file you want and click **OK**.
6. When you save the configuration information, a **.jar** file is generated.

## Adding JMon classes

---

### To add a JMon monitor or event from another Apama project

1. In the Apama Java configuration editor, expand the **Java Application Classes** heading. This displays the current list of the Java classes included in the project.
2. Click **Add**.

The Select Java Apama Files dialog appears showing the available JMon application projects.

3. In the Select Java Apama Files dialog navigate to the class you want to add, select it, and click **OK**.
4. In the Apama Java configuration editor, you can also change the order in which the JMon classes are listed with the **Up** and **Down** buttons. This affects the way the files are ordered in the manifest file and the order in which they are injected in the correlator. Events are injected first, followed by monitors; in both cases, they are injected in the order they are listed here.

## Adding non-JMon Java files

---

### To add non-JMon Java files

1. In the Apama Java configuration editor, expand the **Additional Content** heading. This displays the current list of non-JMon Java files included in the project.

You can add files from other Apama projects and you can add non-project files from outside the Software AG Designer workspace.

2. To add a file from an existing Apama project:
  - a. Click **Add from a project**. The Select a Resource dialog appears.
  - b. Navigate to the file you want to add, select it, and click **OK**.
3. To add a file from outside Software AG Designer:
  - a. Click **Add a Variable**. The Select variables to add to build path dialog appears.
  - b. Select a variable and, if necessary, click **Extend** to identify specific folders on the path. Click **OK**.

## Adding bundles to projects

Bundles are named collections that group the Apama objects that are necessary for different types of applications. *Standard Bundles* contain service monitors and event definitions that are specific to the type of application you are building. For example, applications that include scenarios need the Scenario Service bundle, while applications that use data views need the DataView bundle. *Adapter Bundles* contain adapter configuration files, monitors, and other files needed by the Integration Adapter Framework (IAF). For information on adding adapters, see ["Adding adapters to projects" on page 45](#).

## Bundle instances

Many bundles and adapters contain only EPL or Java files; adding these bundles means that references to the bundle files will be added to the project build path, but does not involve copying any files into the project. However, when you add a bundle that contains customizable files such as `.evt` or adapter configuration files, Software AG Designer physically copies these files into the project, in addition to the changes made to the project build path.

If you add more than one instance of a particular bundle or adapter to the project, Software AG Designer creates separate copies of these files for each instance (except for bundles where it only makes sense to have one copy of the instance files per project). If a bundle contains instance files, you can change and customize those files in any manner. It is possible to add multiple instances of such bundles to a given project, to allow different customization of the same instance files, for example if your application needs to run with two instances of a particular adapter, each connected to a different data provider.

If a bundle contains instance files, the Add Bundle Instance dialog will prompt for a unique *bundle instance name* to identify the instance, to distinguish it from any other instances of the same bundle. This instance name will be included in the filename of the bundle instance files when they are copied into the project. If only one instance of a given bundle will be required, it is fine to use the default bundle instance name. Each bundle or adapter instance is represented in the Project Explorer.

After adding a bundle that contains instance files, the instances can be removed or updated independently (right-click the instance name and select **Remove Instance** from

the pop-up menu). After a bundle with instance files is removed, a dialog will appear to confirm whether you also wish to delete the instance files that were copied to the project. Usually you will want to delete these files at the same time as removing the bundle, as long as backup copies of any important customization have been made.

It is never necessary to have more than one instance of a bundle that does not contain any instance files, so this option is disabled in the Add Bundle Instance dialog.

---

### To add a bundle to an Apama project

1. There are two ways of adding a bundle to a project.
  - *If you are creating a new Apama project, select **File > Project > New > Apama**, give it a name, and click **Next**. The **New Apama Project** dialog opens.*
  - *If you are adding a bundle to an existing project, in the **Project Explorer** right-click the project and select **Apama > Add Bundle**. The Add Bundle dialog opens.*
2. Select the bundle that is appropriate for your application. Click **Finish** or **OK**.

The bundle is added to the `Bundles` folder in your project along with the supporting monitors such as `DatabaseSupport.mon`. If a bundle contains a Correlator Deployment Package you can see this in the bundle's hierarchy in the project's `Bundles` folder in **Project Explorer**.

## Adding adapters to projects

Apama provides standard adapters that communicate with third-party messaging systems, transforming incoming messages into Apama events and, in the opposite direction, transforming Apama events into the proprietary representations required by third-party messaging systems. In addition, the adapter for the Apama Database Connector (ADBC) allows an application to connect to standard ODBC and JDBC data sources as well as Apama Sim data sources. You can add these adapters to an Apama project by the adding the corresponding adapter bundle to the project.

---

### To add an adapter to a project

1. There are several ways of adding an adapter to a project.
  - *If you are creating a new Apama project, select **File > Project > New > Apama**, give it a name, and click **Next**. The **New Apama Project** dialog appears. For more information see, "[Creating Apama projects](#)" on page 26.*
  - *If you are adding an adapter to an existing project:*
    - a. In the **Project Explorer** right-click the project and select **Apama > Add Adapter**. The Add Adapter Instance dialog opens.
    - b. If desired, in the Add Adapter Instance dialog, create a new name for the adapter instance or accept the default instance name. Software AG Designer prevents you from using a name that is already in use.
2. Select the adapter bundle that is appropriate for your application. Click **Finish** or **OK**.

Software AG Designer adds an instance of the adapter to the `Adapters` folder in your project along with the supporting monitors such as `IAFStatusManager.mon` and the associated service monitors. Reference to bundled files associated with the adapters, such as IAF Status Manager and Status Support are listed in the adapter's `Dependent Bundles` folder

You can add multiple instances of a given adapter to a project. This allows you create different configurations of the adapter, for example if your application needs to run with different data providers.

For more information on configuring Apama standard adapters, see "Using Standards Adapters" and "Using the Apama Database Connector" in *Connecting Apama Applications to External Components*.

## Adding Universal Messaging configuration to projects

Universal Messaging (UM) is Software AG's middleware service that delivers data across different networks. It provides messaging functionality without the use of a web server or modifications to firewall policy. In Apama applications, you can configure and use the connectivity provided by UM.

Apama provides the `UM-config.properties` template file in the `etc` folder of your Apama installation directory. The template is for a standard Java properties file. When you add UM configuration to a project, Software AG Designer copies the `UM-config.properties` file to the `config` folder in your project.

---

### To add UM configuration information to a project

1. Right-click the project you want to add UM configuration properties to.
2. Select **Apama > Add UM Configuration**.

A project can contain only one UM configuration properties file. If a `UM-config.properties` file is already in the `config` folder of your project then this option is not available.

3. Right-click the `UM-config.properties` file that is now in the `config` folder of your project. Select **Open With > Apama UM Configuration Editor**.

For details about the content of this file, see "Defining UM properties for Apama applications" in *Connecting Apama Applications to External Components*.

For information about using UM in an Apama application, see "Using Universal Messaging" in *Connecting Apama Applications to External Components*.

## Editing Apama files

---

Software AG Designer provides many features to help you write application code for monitor files (`.mon`), query files (`.qry`), block definition files (`.bdf`), event files (`.evt`), and JMon monitor and JMon event files.

## Obtaining content assistance

To obtain content assistance, enter **Ctrl**+space in a blank line, or after one or more words in a line.

Software AG Designer displays a list of keywords or names that are valid in that location. For example, it might display a list of event types, standard functions, or actions. Double click the one you want or select with the arrow keys and press **Enter**.

To ensure that content assistance is always as up-to-date as possible, turn on automatic builds. Automatic builds ensure that when you add, delete, or change a project resource, Software AG Designer immediately builds the project. Building a project after a resource change ensures that content assistance has access to the most current resources. See ["Build automatically when a resource changes" on page 53](#).

## Using auto-completion

To have Software AG Designer automatically complete the word you are typing, enter **Ctrl**+space.

If you have entered enough characters so that there is only one possible completion, Software AG Designer inserts the rest of the word. If there are two or more possible completions, Software AG Designer displays a list of the completion candidates. Double-click the correct completion or select with the arrow keys and press **Enter**.

Software AG Designer displays only those completion candidates that are valid in the current context and scope.

To ensure that auto-completion is always as up-to-date as possible, turn on automatic builds. Automatic builds ensure that when you add, delete, or change a project resource, Software AG Designer immediately builds the project. Building a project after a resource change ensures that auto-completion has access to the most current resources. See ["Build automatically when a resource changes" on page 53](#).

## Displaying information for events and actions

When you position the mouse cursor to hover over an event declaration or action, a box pops up that displays the event type definition or the signature for the action.

## Specifying comments

To toggle comment notation, click in a line or select one or more lines, and press **Ctrl**+/. Alternatively, you can adjust comment notation.

---

### To adjust comment notation

1. Click in a line or select one or more lines.



2. Right-click anywhere in the code editor.
3. In the menu that appears, select **Source** and then one of the following:
  - **Toggle comment** — Inserts `//` comment notation if the selected line or lines is not a comment. Removes `//` comment notation if the selection is already a comment.
  - **Add block comment** — Inserts `/*` at the beginning of the section, and `*/` at the end of the selection.
  - **Remove block comment** — Removes the `/*` and `*/` notation.

## Using auto-indent

---

To indent one or more lines relative to the entire EPL file

1. Click in a line or select one or more lines.
2. Press **Ctrl+I**.

Alternative: Right-click in the code editor and select **Source > Correct Indentation** .

## Using auto-bracketing

Auto-bracketing is enabled by default. This can be changed in the Apama preferences. See the description of the Apama preferences page ["MonitorScript" on page 154](#) for further information.

When auto-bracketing is enabled, and you enter an opening bracket, brace or quotation mark, a matching closing bracket, brace or closing quotation mark is automatically inserted.

## Using tabs

By default, tabs are inserted in the EPL editor when you press the TAB key.

This is defined in the Apama preferences. If you want to insert spaces instead, see the description of the Apama preference page ["Editor Formatting" on page 155](#).

## Defining shorthand (templates) for frequently used EPL code

EPL templates provide a way for you to define a short name for a longer pattern that you often specify in an EPL file. For example, Apama provides the `dict` template. When you enter `dict` followed by a space, `dictionary<>` is automatically inserted.

Apama provides some templates in Software AG Designer and you can define additional templates. Each Apama project can use all EPL templates.

The EPL templates are defined in the Apama preferences. For more information, see the description of the Apama preference page ["Editor Templates" on page 156](#).



## Sharing templates among Software AG Designer installations

You can define templates in one Software AG Designer installation, and export them for use in another Software AG Designer installation.

### To export templates and import them in another installation

1. In the source Apama project, define all the templates that you want to share. See ["Defining shorthand \(templates\) for frequently used EPL code" on page 48](#).
2. Go to the Apama preferences and export them all. For more information, see the description of the Apama preference page ["Editor Templates" on page 156](#).
3. Go to some other Software AG Designer installation.
4. Go to the Apama preferences of this installation and import the templates. For more information, see the description of the Apama preference page ["Editor Templates" on page 156](#).

## Specifying colors to distinguish EPL elements

In EPL files, the following colors are used by default to indicate the various parts of EPL code:

- Keywords are dark magenta.
- Types are red.
- Comments are green.
- Literal values are blue.
- Operators (+, -, =, and so on) are black.
- All other text is black.

These colors are defined in the Apama preferences. If you want to use different colors, see the description of the Apama preference page ["Editor Colors" on page 155](#).

## Shortcuts when editing Apama files

Software AG Designer provides the following shortcuts, in addition to the usual Eclipse shortcuts:

Action	Key	Description
Auto-completion and content assistance	<b>Ctrl</b> +space	If the text you entered has only one possible completion, Software AG Designer inserts the completion. If you did not enter any text, or if there are two

Action	Key	Description
		or more completion candidates, Software AG Designer displays candidates for you to choose from.
Toggle comment line notation	<b>Ctrl+/ </b>	Toggles comment notation for selected lines. That is, this action inserts or removes // from the beginning of each line.
Insert block comment	<b>Ctrl+Shift+/ </b>	Makes the selected text a block comment. That is, this action inserts /* at the beginning of the selected text, and */ at the end of the selected text.
Remove block comment notation	<b>Ctrl+Shift+\</b>	Removes block comment notation from the selected text. That is, this action removes /* from the beginning of the selected text, and */ from the end of the selected text.
Auto-indent	<b>Ctrl+I</b>	Inserts appropriate indents in selected lines relative to the entire file.
Move line (s) up	<b>Alt+up arrow</b>	Move the selected line or lines to be before the previous line.
Move line(s) down	<b>Alt+down arrow</b>	Move the selected line or lines to be after the subsequent line.
Shorthand for any EPL	Templates	Templates let you define short names for patterns you frequently specify in EPL files. See <a href="#">"Defining shorthand (templates) for frequently used EPL code" on page 48.</a>

## Navigating in Apama files

---

In addition to the usual Eclipse navigation tools, Software AG Designer lets you navigate to particular parts of an EPL file as described in the topics below.

You can use these features to navigate the code of EPL files (`.mon`), and block definition files (`.bdf`).

### Using the Outline view to navigate

In an Apama perspective, the Outline view displays the event types and monitors in the currently open `.mon` or `.bdf` file, showing event parameters and actions and monitor fields, actions, and listeners within actions. When you click an entry in the Outline view, the focus in the editor view jumps to the code that defines the item you clicked.

Software AG Designer updates the Outline view each time it saves your EPL file. If you add a resource to your EPL file but you do not save your file, the new resource does not appear in the Outline view until you save your file.

### Using the Quick Outline to navigate

While you are viewing a `.mon` or `.bdf` file in the editor, you can use the Quick Outline to jump to an event type, monitor, or action. Press **Ctrl+O** or select **Navigate > Quick Outline** to display the Quick Outline. Software AG Designer pops up a list of event parameters and actions and monitor fields, actions, and listeners within actions for the current EPL file. Click an entry in the list to jump to the code that defines the item you click.

## Jumping to an event or action definition or variable declaration

---





### To jump to an event or action definition or variable declaration

1. Select a reference to the event, action, or variable whose declaration you want to view.
2. Press **F3** or choose **Navigate > Open Declaration** from the Software AG Designer menu bar. Software AG Designer highlights the line that begins the declaration for the selected event or variable.

## Searching in EPL files

You can search for event types in your project's `.mon` or `.bdf` files.






### To start a search

1. In the Developer or Workbench perspective, select **Search > Search** from the menu bar. In the Developer perspective, you can also click the Search icon . The Search dialog appears. Make sure the **MonitorScript Search** tab is selected.
2. Type the event type you want to search for in the **Event name ( fully qualified )** field. You can use wildcard characters and regular expressions in your search. Modify the search details as necessary, for example by specifying if the search should be case sensitive or if you want to limit the scope of the search. You can limit your search to templates (event listeners), senders (route, enqueue, and emit), and variables (event types are used as variables).
3. Click **Search**. Results are displayed in the Search view (**Search** tab), showing the files where the search term occurs and how many times it occurs. To display the first occurrence of the search term, double click the file name or click the **Show Next Match**  button. This opens the file containing the term in the appropriate editor. You can navigate through all occurrences of the search term with **Show Next Match**  and **Show Previous Match**  buttons.

## Building Apama projects

In Software AG Designer, “building a project” refers to the process in which the project is validated, and any errors or warnings flagged to the user. This section describes the features Software AG Designer provides to control when Software AG Designer will build an Apama project.

Software AG Designer tries to complete each build. It does not stop when it finds an error. If Software AG Designer finds problems during validation, it indicates them as follows:

- In the Problems view, the error icon  appears at the beginning of each line that describes an error.
- In the Project view, the error icon  appears at the beginning of the name of a project that contains at least one file that is not valid.
- In the Project view, the error icon  also appears at the beginning of the name of each file that is not valid.
- In the file editor, the error icon  appears at the beginning of each line that contains an error.
- In the overview scroll bar to the right of the editing pane, regardless of how long the file is, a mark  appears for every error in the file. You can hover over a mark to view a description of the error that it flags, or you can click on any mark to display the line that contains that error.

## Build automatically when a resource changes

Software AG Designer can automatically build a project whenever you add, delete, or change a resource in that project. This is the default and Apama strongly recommends that you leave this feature on. Building a project automatically ensures that content assistance menus are always up to date.

---

### To toggle Build Automatically on or off

1. Select **Project** in the menu.
2. In the drop-down menu that appears, if a checkmark appears in front of **Build Automatically**, automatic builds are already turned on. Otherwise, select **Build Automatically** to turn it on.

## Build all Apama projects

To validate all Apama projects in your workspace, choose **Project > Build All** from the menu bar.

## Build one Apama project

---

### To validate one Apama project

1. In the Project Explorer view, select the project you want to build.
2. In the menu bar, choose **Project > Build Project**.

## Build a working set

When an Apama project is large, you might find it more efficient to build a subset of the project rather than the entire project. To build a subset of an Apama project, specify a working set that contains the files you want to build. Then build the working set you defined.

---

### To define a working set

1. In the Software AG Designer menu bar, choose **Project > Build Working Set > Select Working Set**.
2. In the Select Working Set dialog, click **New**.
3. Double-click **Resource**.
4. Specify a name for this working set.
5. Expand the project(s) that contains the file(s) that you want to be in the working set.
6. Select the file(s) that you want to be in the working set, and click **Finish**.

7. Back in the Select Working Set dialog, select the working set you just defined and click **OK**.

To build this working set, choose **Project > Build Working Set > *working\_set\_name*** from the menu bar.

## Clean and rebuild projects

When Software AG Designer cleans a project, it discards all build problems and build states.

---

### To rebuild a project from scratch, as though you have never built it before

1. In the Software AG Designer menu bar, choose **Project > Clean**.
2. Indicate which projects you want to rebuild from scratch or select **Clean all projects**.
3. Click **OK**.

## Configuring the project build path

By default, Software AG Designer will include all of the files in a project when building (and launching) it. However, for many applications it will be necessary to customize the build path, adding additional files from outside the project (and less commonly, limiting the set of files under the project directory that will be included). The set of files under the project directory that will be build can be customized using the Source tab. Software AG Designer provides three ways to add additional files from outside the project to its build path:

- Required Projects
- External MonitorScript Dependencies
- Bundles

## Project source files

By default, all files in the project directory are included in the build path, but this can be customized. Files within a project are grouped into folders and you can specify which folders should be included when a project is built. In addition, within each folder you can specify patterns that determine which files should be included or excluded.

---

### To specify files this way

1. Select the project in the Project view and select **Project Properties** from the Software AG Designer menu.
2. Expand **Apama**.
3. In the Properties dialog, select **MonitorScript Build Path**, and click **Source**.

4. On the Source tab you add new folders and specify which files to include in the build process. For example, you can include all event files with the `.evt` extension. For folders in the project, you can also modify which files to include or exclude.

## Specifying projects

A project can make use of other Software AG Designer projects.

---

### To specify that your project should include another project in the build process

1. Select the project that requires another project in the **Project** view and select **Project > Properties** from the Software AG Designer menu.
2. Expand **Apama**.
3. In the Properties dialog, select **MonitorScript Build Path**, and click **Projects**.
4. On the Projects tab, click **Add** to display the Required Project Selection dialog where you specify which projects to include in the build process.

## Specifying external dependencies

When an EPL file in your project depends on an external EPL file, you must explicitly specify the dependency. For example, if you refer to event types or actions that are defined in other EPL files, you must specify the files that define those constructs. This ensures that Software AG Designer includes the external file in the validated project. It also ensures that the Software AG Designer builder, content assistance facility, and launcher have access to the most up-to-date version of each required file.

How you define dependencies depends on whether you are the only one using your project or you share the project with one or more users.

### *Specifying dependencies for a single-user project*

When you are the only one using a project, specify a file that your EPL file depends on.

---

### To specify the file

1. Select the project for which you want to define a dependency.
2. In the Software AG Designer menu bar, choose **Project > Properties**.
3. In the Properties dialog, select **Apama**.
4. Click **MonitorScriptBuild Path**.
5. Display the External Dependencies tab.
6. Click **Add External**.
7. In the MonitorScript File Selection dialog, navigate to the file your project requires.
8. Double-click the file.
9. Click **OK** in the Properties dialog to specify the dependency.

### ***Specifying dependencies for a multi-user project***

When Software AG Designer builds a project, it defines any dependencies in an XML file that hardcodes the path to the external file(s). This works correctly if you do not share projects among multiple users. However, if two or more users share a project, they might not store external files in the same location.

If you are sharing projects among two or more users, you should use a variable to define an external file on which your EPL file depends. For example, Software AG Designer automatically defines `APAMA_HOME` and `APAMA_WORK` variables that can be used to locate files under either of those directories in a way that is independent of exactly where Apama has been installed on the current machine.

---

#### **To use a variable to define a dependency**

1. Select the project, or a file in the project, for which you want to define a dependency.
2. In the Software AG Designer menu, choose **Project > Properties**.
3. In the Properties dialog that appears, expand **Apama**.
4. Click **MonitorScript Build Path**.
5. Display the External Dependencies tab.
6. Click **Add Variable**.
7. In the New Variable Dependency Entry dialog, do one of the following:
  - If a variable already exists that identifies the required file, double-click that variable. Then click **OK**. You are done and can skip the remaining steps.
  - To define a new variable that identifies the required file, click **Configure Variables**.
8. In the New Variable Entry dialog, in the **Name** field, enter the name of the new variable. The convention is to use `UPPER_CASE` names for build path variables (for example, `APAMA_HOME`).
9. Click **File...** or **Folder...** according to which one you want the variable to represent.
10. Navigate to the appropriate folder or EPL file and double-click it.
11. Click **OK** twice.
12. Click **OK** in the Preferences dialog; Software AG Designer confirms that the classpath variables have changed, and prompts you to indicate whether you want to rebuild your project so that it can use the new variable. Click **Yes**.

### **Defining MonitorScript Build Path variables**

In any Apama project, you can define a variable that you can use in all your Apama projects. You might find it useful to define a variable in a situation where multiple users share the same project. If the project is dependent on one or more external files, not all users might have the external file stored in the same location. Each user defines a variable to specify the location of a shared file dependency.



You define an Apama project variable in the Apama preferences. For more information, see the description of the Apama preference page ["MonitorScript Path Variables" on page 157](#).

## Importing projects

---

**To import an Apama project created on another machine into this Software AG Designer workspace**

1. Select **File > Import > General > Existing Projects into Workspace** from the Software AG Designer menu. Click **Next**.
2. In **Select root directory** specify the root directory where the project is located.
3. Add a check box next to the project(s) you want to import.
4. Check the **Copy projects into workspace** checkbox if you want to copy all the project files into the workspace directory (located under `APAMA_WORK`)  
Leave the check box unchecked if you want to simply link to the project in its current location instead.

If you have an Apama application that is not currently part of an Apama project:

1. Select **File > New > Apama Project** from the Software AG Designer menu.
2. Enter the name of the new project.
3. Uncheck the **Use default location** check box.
4. Specify the folder containing the files that you wish to import as a new project.
5. Click **Finish**.

A project that you import might have dependencies on environment variables, bundles, blocks, or functions that have not yet been added to Software AG Designer. As an alternative to explicitly adding each dependency, see ["Setting up the environment before importing projects" on page 58](#).

## Importing adapter configurations

---

You can import a Web service client adapter configuration from an archive file that has been generated in Software AG Designer.

**To import a Web service client adapter configuration into an Apama project**

1. In the Project Explorer view, right-click the name of the project and select **Import** from the pop-up menu. This displays the Import dialog.
2. In the Import dialog, expand **Software AG**.
3. Select **Apama Adapter Configurations** and click **Next**. The Adapter Configurations Import Wizard appears.

4. In the Adapter Configurations Import Wizard, specify the archive file that contains the adapter configuration you want to import. You can use the **Browse** button to navigate to the desired project.
5. Accept or specify the name of the project into which you want to import the adapter configuration. You can use the **Browse** button to navigate to the desired location.
6. Click **Finish**. This adds the adapter configuration to the specified Apama project.

## Setting up the environment before importing projects

---

A project that you want to import into Software AG Designer might have dependencies on any of the following:

- Environment variables
- Catalogs of blocks, bundles or functions
- String substitutions

Before you can build your project, you would need to add each dependency to Software AG Designer. An alternative to adding each dependency is to define the dependencies in a file, and place the file in the `software_ag_install_dir\Designer\extensions` folder. When Software AG Designer starts, it collects any files in its `extensions` folder and uses them to set up the Software AG Designer environment. When you then import your project its dependencies will already be in place.

### Format of extensions file

A file in the `extensions` folder must have the `.ste` (Studio Tuning Extension) extension and the data it contains must be in the following format:

- Define each item to be added to Software AG Designer on its own line.
- The first value in each line must be the type of the item you want to add. The type must be one of the following:
  - `BLOCK_CATALOG`
  - `BUNDLE_CATALOG`
  - `FUNCTION_CATALOG`
  - `STRING_SUBSTITUTION`
  - `VARIABLE`
- In each line, insert a semicolon between values.
- Insert `#` at the beginning of a line to make it a comment.
- The values you specify vary according to the specified type. Path specifications must be fully qualified; they cannot be relative. In a path specification, you can use a

variable, which can be defined directly in Software AG Designer or in any `.ste` file. You can specify the items in any order.

- For block, bundle, or function catalogs, the format is the type followed by a path. For example:

```
BLOCK_CATALOG ; C:\Program Files\MyCompany\MyApp\MyBlockCatalog
```

- For string substitutions, the substitution value cannot be edited or removed in Software AG Designer. For details, See **Java development user guide > Reference > Preferences > Run/Debug > String Substitutions** in the Eclipse help provided with Software AG Designer. The format is as follows:

```
STRING_SUBSTITUTION ; variable_name ; path ; description
```

For example:

```
STRING_SUBSTITUTION ; HOME ; C:\MyApp ; Install dir for my app
```

- For variables, the variable's value cannot be edited and the variable cannot be removed in Software AG Designer. The format is as follows:

```
VARIABLE ; variable_name ; path
```

For example:

```
VARIABLE ; HOME ; C:\MyApp
```

Suppose you have a `.ste` file in place and you start Software AG Designer. If you subsequently modify the content of that `.ste` file you must restart Software AG Designer for the changes to take effect.

### Results of using an extensions file

After you define an extensions file, place it in the `software_ag_install_dir\Designer\extensions` folder and then start Software AG Designer.

You should see the items you defined in the extensions file in the appropriate Software AG Designer dialogs. For example, select **Window > Preferences** and then expand **Software AG > Apama** and click **Catalogs**. Any catalogs you specified in the extensions file should appear in the appropriate tab. However, you cannot edit or remove any catalogs that were added to Software AG Designer by means of an extensions file.

If you import a project that uses any of the items specified in the extensions file then the imported project will be valid with regard to any of these dependencies.

## Exporting project information

You can export a project's initialization file list, deployment script, scenarios, Correlator Deployment Packages, and adapter configurations.

## Exporting a project initialization file list

---

### To export a project initialization file list

1. Select **File > Export** from the Software AG Designer menu.
2. Expand **Software AG**.
3. Select **Project Initialization File List** and click **Next**.
4. If you want the file list to also contain all event files from the project, select **Include event files (\*.evt)**.
5. Specify a name for the file to hold the initialization file list and click **Finish**.

This creates an ordered list of the files on the project build path and saves it in a text file (with one entry per line). This file can be imported into an Apama component's Initialization tab in the Enterprise Management and Monitoring console (EMM) to help convert a development Apama project into a production EMM deployment.

Note that at present this list does not include any scenario (`.sdf`) files on the build path. Scenarios should therefore be exported to an EPL file manually (using Event Modeler), and the EPL files added directly to EMM's initialization list once the Apama Initialization File has been imported.

## Exporting to a deployment script

You can export an application's launch configuration to create a deployment script. This generates the build files, configuration files, property definition files, scripts, and EPL files from scenarios and copies other resources such as dashboards that are used by Ant to build and launch the project on a different machine.

---

### To export an Apama launch configuration to a deployment script

1. In the Project Explorer view, right-click the name of the project and select **Export** from the pop-up menu. This displays the Export dialog.
2. In the Export dialog, expand **Software AG**.
3. Select **Apama Ant Export** and click **Next**. The Ant Export dialog appears.
4. In the Ant Export dialog, specify the settings as follows:
  - **Launch configuration** — The export operation uses the project's default launch configuration, but if a project has multiple launch configurations, you can select which one to export. If you want to select a different launch configuration, click **Browse**. This will display the Choose Launch Configuration dialog.
  - **Destination directory** — The name of the directory for the exported files.

- **Generate initialization list during launch** — Dynamically creates the file injection list from the project directory when the exported deployment script is executed, rather than during the export process.

When you select this option you need to export your project's launch configuration only once. The generated scripts specify the location of your project directory and then use the content of your project directory to create the file injection list at deployment time. If you do not select this option, then you must re-export the configuration each time you add, remove or edit a file in your application.

For JMon applications that you develop in Software AG Designer, Software AG Designer creates the required `.jar` files whenever you modify your Java files. If you do not develop your JMon application in Software AG Designer, see "Generating deployment descriptor files from annotations" in *Developing Apama Applications* for information about building the `.jar` file for your application. Ensure that your application's `.jar` file is in your project directory before export.

If you selected a shared location when you created the launch configuration that you are exporting then Software AG Designer generated two files that contain the launch information (`.deploy` and `.launch`) and put them in the specified shared location. After you create the launch configuration, any changes you make to the launch configuration are reflected in these files. Since the exported deployment script uses these files at deployment time, any launch configuration changes will also be reflected upon deployment. Except, if you change the shared location then you must re-export the launch configuration to a new Ant deployment script. If you do not, the old Ant deployment script fails because it cannot find the `.deploy` file.

When **Generate initialization list during launch** is selected the option to **Generate Scenario EPL during export** is not available. Instead, EPL `.mon` files are always created from scenario files upon deployment. Also, **Copy resources to destination** is not available because the script points to the project directory.

- **Copy resources to destination** — By default, the export operation copies only the project's dashboard definitions. To change the specific resources that Software AG Designer exports, click **Browse** to display the Export Resource Browser dialog and specify the resources you want.

For an Apama query application, if you are not selecting **Generate initialization list during launch**, then you will need to use the Apama macros for Ant to inject the query application files. See the `apama-macros.xml` file in the `etc` folder of your Apama installation directory.

- **Use relative paths** — By default, the generated `build.xml` file uses relative pathnames for the application's monitors, events, scenarios, jars, and adapter configurations and properties. Uncheck this box if you want to use absolute pathnames.
- **Include custom file** — If you want the exported launch configuration to perform other operations, select this option to generate a stub `custom.xml` file. The

`custom.xml` file has `pre-custom` and `post-custom` targets where you can add the desired operations.

- **Batch resources when possible** — By default, the `build.xml` file generated by the export operation specifies that all monitors in the project will be injected in a batch when the application is launched. If you want to inject each monitor separately when the application is launched, uncheck this check box.
- **Launch correlator in separate console** — By default, the exported launch configuration will launch the correlator in a separate console. Uncheck this check box if you want to start the correlator in the console where the launch is started.
- **Generate Windows scripts** — Exports all scripts used by the launch configuration in Windows form.
- **Generate Unix scripts** — Exports all scripts used by the launch configuration in UNIX form.
- **Generate Scenario EPL during export** — Generates EPL files for scenarios during the export operation. If you do not select **Generate initialization list during launch** and you want to copy the scenario definition files instead of generating EPL files, uncheck this check box. Note that when you select **Generate initialization list during launch** any scenarios are converted to EPL `.mon` files when the launch configuration injects the files into the correlator. Consequently, this option is not available when **Generate initialization list during launch** is selected.
- **Generate export log** — By default the export operation generates a log file that records the export operation. Uncheck the check box if you do not want to record the log file.

Click **Finish**. The files are generated in the specified destination directory.

## Exporting scenarios

From Software AG Designer's Export dialog, you can export a project's scenarios in two ways:

- As EPL code — See "Exporting scenarios as EPL" in *Developing Apama Applications*.
- As block templates — See "Exporting scenarios as block templates" in *Developing Apama Applications*.

## Exporting Correlator Deployment Packages

You can export a project's EPL (`.mon` and `.qry` files) and scenario files to a Correlator Deployment Package (CDP). CDP files use a proprietary, non-plaintext format that treats EPL and scenario files in a manner similar to the way a JAR file treats a collection of Java files.

**Note:** When you upgrade to a new major version of the product, you must regenerate any CDP that was created by exporting query files or scenario definitions from the previous version.

---

### To export CDP

1. In the Project Explorer view, right-click the name of the project and select **Export** from the pop-up menu. This displays the Export dialog.
2. In the Export dialog, expand **Software AG**.
3. Select **Export as Correlator Deployment Package** and click **Next**. The Correlator Deployment Package wizard appears.
4. On the first page of the Correlator Deployment Package wizard, if necessary specify a project in the **Project** field. You can use the **Browse** button to navigate to the desired project.
5. Also on the first page of the Correlator Deployment Package wizard, in the **Package Filename** field, specify the name of the CDP file you want to create. You can use the **Browse** button to navigate to the desired location.
6. Click **Next**. The second page of the Correlator Deployment Package wizard appears.
7. On the second page of the Correlator Deployment Package wizard, specify the files you want to add to the package as follows:
  - a. On the **Artifacts Selection** tab, select the EPL and scenario files you want to include in the package.
  - b. On the **Injection Order** tab, use the **Move Up** and **Move Down** buttons to specify the order in which you want to inject the EPL and scenario files.
8. Click **Finish**.

## Exporting adapter configurations

You can export a project's Web service client adapter configuration to an archive file, which you can generate in Software AG Designer.

---

### To export a Web service client adapter configuration from an Apama project

1. In the Project Explorer view, right-click the name of the project and select **Export** from the pop-up menu. This displays the Export dialog.
2. In the Export dialog, expand **Software AG**.
3. Select **Apama Adapter Configurations** and click **Next**. The Adapter Configurations Export Wizard appears.
4. In the Adapter Configurations Export Wizard, if necessary, specify a project in the **Project** field. You can use the **Browse** button to navigate to the desired project.

5. In the **Adapter Configurations** field, select the adapter configurations you want to export.
6. In the **Adapter Resources** field, select the adapter resources you want to export.
7. In the **To archive file** field, specify the archive file that you want to contain the exported adapter configuration. You can use the **Browse** button to navigate to the desired location.
8. Click **Finish**. This creates an archive file in the specified location. You can import this archive into any Apama project.

## Exporting ApamaDoc

In Software AG Designer, you can use the ApamaDoc tool to generate reference documentation for the EPL source code you add to a project. From the Export dialog, select **Software AG > ApamaDoc Export**. This generates static HTML pages that document the structure of all EPL code in a project.

For detailed information on how to annotate your EPL source code and generate ApamaDoc, see "Generating documentation for your EPL code" in *Developing Apama Applications*.

## Deleting projects and resources

---

If you want to delete projects or resources you should use Software AG Designer to do so, rather trying to delete them directly from the file system.

### Deleting resources

---

#### To delete a resource from a project

1. In the Project Explorer view or Workbench Project view, right-click the resource and select **Delete** from the pop-up menu or select the resource and select **Edit > Delete** from the menu. In the Workbench Project view you can select the resource and click the Delete button.
2. In the Confirm Resource Delete dialog, click **Yes** if you want to proceed.

### Deleting projects

---

#### To delete a project

1. In the Project Explorer view or Workbench Project view, right-click the project and select **Delete** from the pop-up menu or select the resource and select **Edit > Delete** from the menu. In the Workbench Project view you can select the resource and click the **Delete** button.



2. In the Confirm Project Delete dialog select whether or not you want to delete all the project's resources from the file system, or simply remove the project from the **Project** view leaving all files in place. After selecting the latter option, such projects can be added to the workspace again using the Import wizard (see ["Importing projects" on page 57](#)). In most cases it is more useful to select the option to delete the project's contents at the same time as the project itself, to avoid confusion.

## Adding the Apama nature to a project

---

If you are working with a project that is not an Apama project, for example, a Java project, you can apply the Apama nature to the project with Software AG Designer. The project then becomes an Apama project in addition to whatever natures it had before.

### To add the Apama nature to a project

---

1. In the Project Explorer view (or in the Package Explorer view if the project is a Java project), right-click the project.
2. Select **Apama > Add Apama Nature** from the pop-up menu.

After you apply the Apama nature to a project, the project shares all the features of any other Apama project, for example, EPL errors will be detected and flagged, and the project can be launched in an Apama correlator.

Note that the **Add Apama Nature** menu item is not available if a project is already an Apama project.

## Adding the Java nature to an Apama project

---

You can add two different types of Java nature to an Apama project:

- **Java nature.** The Java nature applies standard Eclipse Java support to an Apama project, so that all operations that can be performed on a normal Java project can also be performed on the Apama project.

When a project has the Java nature, Software AG Designer:

- Uses `ApamaJavaLibrary` to add all necessary JAR files to the project's Java build path.
  - Creates and updates the `jmon-jar.xml` file. This is the deployment descriptor file required by each JMon application. Inside the correlator, the JVM processes the deployment descriptor file and uses it as a guide to the event types and monitor classes to load.
  - Generates and maintains your application's JMon JAR file in the `project_name java application files` folder of your project.
- **Apama Java nature.** The Apama Java nature enables the Apama project to use Apama's Java API and adds support for various operations.

When you create a new Apama project, you can add these natures by selecting the corresponding options. See ["Creating Apama projects" on page 26](#) for further information.

If you want to add these natures to an existing Apama project, proceed as described below.

**Note:** The terms "Java nature" and "Java support" are used interchangeably in this documentation. Both refer to the same functionality.

#### To add the Java or Apama Java nature to an existing project


1. In the **Project Explorer**, right-click the Apama project.
2. Click either **Apama > Add Java Nature** or **Apama > Add Apama Java Nature**.

## Viewing all EPL objects in all projects

The **EPL Objects** view shows all the EPL objects - the Apama events, monitors and queries - defined in the projects in your Software AG Designer workspace. These objects are the events, monitors and queries that you define, as well the events and monitors defined in the Apama bundles that are included in your projects.

If the **EPL Objects** view is not displayed, from the Software AG Designer menu select **Window > Show View > Other > Apama > EPL Objects**.

Double-click the name of an object in the **EPL Objects** view, the EPL file that defines the object appears.

The **EPL Objects** view organizes the display of EPL objects by project. You can display events, monitors, and/or queries. To specify what type of objects you want to display, click the **View Menu** icon on the view's title bar (  ) and from the drop-down menu, select **Customize View** and then either **Events**, **Monitors**, **Queries** or **Show All**.

You can display the EPL objects in a project grouped by package name (**Hierarchy**):

You can also display the EPL objects in a project alphabetically (**Flat**):

To change the way the object names are displayed, click the **View Menu** icon on the view's title bar and from the drop-down menu, select **Group By** and then either **Hierarchy** or **Flat**.

You can also specify a regular expression in the text box to filter the EPL entries shown in the view. For example, entering `*scenario` in the text box results in showing only entries with "scenario" in their names.

Click on the **Refresh** icon in the view's title (  ) bar to reload the entries in the **EPL Objects** view.

## Internationalizing Apama applications

---

By default, Software AG Designer saves Apama project files in your platform's native encoding. If you choose to save Apama project files in UTF-8 encoding in the **Resource** tab of a file or a folder's Properties dialog, Software AG Designer adds a UTF-8 BOM character at the beginning of each file. This indicates that the contents are in UTF-8. The character is required to be compatible with other Apama tools.

---

### To specify the encoding for Apama projects

1. In the **Apama Project Navigator** view, click the project for which you want to define the encoding.
2. In the Software AG Designer menu bar, choose **Project > Properties**.
3. In the Properties dialog, click **Info**.
4. In the **Text File Encoding** group box, click **Other** and select the encoding you want.
5. Click **OK**.

## Checking the error log

---

While using Apama with Software AG Designer, you might receive a message that prompts you to check the error log. If the **Error Log** view does not already appear with the **Problems**, **Tasks**, and **Console** views, display it as described below.

See the `APAMA_WORK\logs\apama_designer.log` file for more detailed information about any unexpected problems you encounter while using Apama with Software AG Designer.

---

### To display the Error Log view

1. From the **Window** menu, choose **Show View > Other**.
2. In the Show View dialog, expand **General**.
3. Double-click **Error Log**.

The **Error Log** view is now shown.

## Using Software AG Designer to configure adapters that use UM

### To configure an IAF adapter to use Universal Messaging

1. Add support for UM to your project. See ["Adding Universal Messaging configuration to projects" on page 46](#).
2. Add one of the following adapters to your project:
  - File Adapter
  - JDBC Adapter
  - ODBC Adapter
  - Sim File Adapter
  - WebServices Client Adapter
3. Open the instance of the adapter you just added. In the **Settings** tab, you can see the following substitution variables:
  - APAMA\_MSG\_ENABLED
  - UM\_MSG\_ENABLED

Use Software AG Designer's launch configuration editor to manage these substitution variables.
4. Select **Run > Run Configurations...**
5. In the **Run Configurations** dialog, select the Apama project that contains the adapter you want to configure and then click the **Components** tab.
6. In the **Components** tab, double-click the adapter you want to configure.
 

In the **Adapter Configuration** dialog, if you added UM support to your project, the default for a new adapter in your project is that the **UM Messaging** radio button is selected. Adding UM support to a project does not change an existing adapter configuration.
7. Ensure that **UM Messaging** is selected. This sets the related substitution variables as follows:
  - APAMA\_MSG\_ENABLED is false.
  - UM\_MSG\_ENABLED is true.
8. Click **OK** , **Apply**, and then **Close** to save the adapter configuration in that launch configuration.

Software AG Designer manages the APAMA\_MSG\_ENABLED and UM\_MSG\_ENABLED substitution variables by means of the launch configuration.

If you click the **XML Source** tab you can see that the `<apama>` element and the `<universal-messaging>` element each contain the `enabled` attribute. In the `<apama>` element, Software AG Designer sets the `enabled` attribute to the `@APAMA_MSG_ENABLED@` substitution value. In the `<universal-messaging>` element, Software AG Designer sets the `enabled` attribute to the `@UM_MSG_ENABLED@` substitution value. When you launch the project, Software AG Designer uses the settings of the `APAMA_MSG_ENABLED` and `UM_MSG_ENABLED` substitution variables to set the value of the `enabled` attribute in each element.

The default is that the `enabled` attribute is set to `true`. Consequently, if you delete the `enabled` attribute, it is as if it is set to `true`.

See also: "Configuring adapters to use UM" in *Connecting Apama Applications to External Components*.



# 3

## Using Query Designer

---

■ Use cases for queries .....	72
■ Adding query files to projects .....	73
■ Steps for using Software AG Designer to implement queries .....	74
■ Overview of creating a query in the Design tab .....	74
■ Creating queries in Query Designer .....	75
■ Configuring query projects .....	95
■ Exporting query deployment scripts .....	95

Apama's Query Designer editor provides a graphical environment that business analysts can use to define and update Apama queries without the need to write source code. An Apama query monitors a very large number of real-world entities and processes events on a per-entity basis, for example, all events related to one credit card account.

**Note:** EPL monitors and EPL queries (also referred to as Apama queries) provide two different approaches to implementing an Apama application. See "Comparison of queries and monitors" in *Introduction to Apama*.

Query Designer provides two views of a query definition:

- **Design** — Business analysts define and update query definitions in this view. The **Design** tab provides an event palette and a canvas for defining an event pattern of interest. Toolbars, tooltips, and dialogs make it easy to define the events of interest to the query as well as any required parameters, conditions for finding matches, aggregate calculations and actions to take when a match set is found.
- **Source** — Application developers can examine and update query definition source code in the EPL code editor available from the **Source** tab.

Query Designer keeps the two views synchronized. Any changes made in the **Design** view are reflected in the **Source** view and vice versa.

Apama provides several sample query applications, which you can find in the `samples\queries` directory of your Apama installation directory.

See also: "Defining Queries", which describes the EPL constructs for writing query code, in *Developing Apama Applications* and "Deploying and Managing Queries" in *Deploying and Managing Apama Applications*.

## Use cases for queries

---

Apama queries are useful when you want to monitor incoming events that provide information updates about a very large set of real-world entities such as credit cards, bank accounts, cell phones. Typically, you want to independently examine the set of events associated with each entity, that is, all events related to a particular credit card account, bank account, or cell phone. A query application operates on a huge number of independent sets with a relatively small number of events in each set.

One use case for Apama queries is to detect subsequent withdrawals from the same bank account but from locations that make it improbable that the withdrawals are legitimate. Very large numbers of withdrawal events would stream into your application. A query can segregate the transactions for each bank account from the transactions of any other bank account. Your query application can then check the transaction events for a particular account to determine if there have been withdrawals within, for example, a two-hour period from locations that are more than two hours apart. You can write a query application so that if it finds this situation the response is to contact the credit card holder.



Another use case is to detect repeated maximum withdrawals from the same automatic teller machine (ATM) within a short period of time. This might be due to a criminal with a stack of copied cards and identification numbers. In this case, a query can segregate events by ATMs. That is, the transactions conducted at a particular ATM would be in their own partition, separate from transactions conducted at any other ATM. Your query application can check the events in each partition to determine if, for example, there are repeated withdrawals of \$500 within one hour. If such a situation is found your query can be written to send an alert message to the local police.

Another use case for Apama queries is to offer a better data plan to new smartphone users. Large numbers of events related to cell phone customers would come into the system. Your query application can create sets of events where each set, or partition, contains the events related to one cell phone customer. When your query detects an upgrade from a flip phone to a smart phone, your application can automatically send a message to that customer that outlines a better data plan.

In summary, the characteristics of an Apama query application include:

- You want to monitor a very large number of real-world entities.
- You want to process events on a per-entity basis, for example, all events related to one credit card account.
- The data you need to retain in order to run Apama queries is either too large to fit on to a single machine or there is a requirement to place it in shared, fast-access storage (a cache) to support resilience/availability requirements.

More information about the use cases for queries can be found in "Understanding queries" in *Introduction to Apama*.

## Adding query files to projects

---

Apama queries are useful when you want to monitor incoming events that provide information updates about a very large set of real-world entities such as credit cards, bank accounts, cell phones. Typically, you want to independently examine the set of events associated with each entity, that is, all events related to a particular credit card account, bank account, or cell phone. A query application operates on a huge number of independent sets with a relatively small number of events in each set.

---

### To add a query file to a project

1. In Project Explorer, right-click the project you want to add a query to and select **New > Query File**.
2. In the **New Apama Query File** dialog, the **Containing folder** field shows the name of the project you selected. Accept this or enter the name of the project you want to add the query to.
3. In the **File name** field, enter a name for the query file.

4. In the **Query name** field, when you enter the file name in the **File name** field, the same file name appears as a query name in the **Query name** field. You can either accept or edit the query name.
5. In the **Package** field, you can optionally specify the EPL package that contains this query. If you do not specify a package then the query file is in the default package which contains EPL files that do not specify a package name.
6. Click **Finish**.

The new query file is added to the project's **queries** folder and the new query is opened in the **Design** tab of the Query Designer. Also, the `Queries` Support bundle is automatically added to your project, if your project does already contain this bundle. A query application project must contain the `Queries` Support bundle. After you add a query file to a project, you can view it along with the other EPL objects in your project. See ["Viewing all EPL objects in all projects" on page 66](#).

## Steps for using Software AG Designer to implement queries

### To use Software AG Designer to implement a query application

1. Create an Apama project for your query application. See ["Creating Apama projects" on page 26](#).
2. Define the events you want your query to operate on. See ["Creating new event definition files for EPL applications" on page 28](#).
3. Add a query file to your project. See ["Adding query files to projects" on page 73](#).
4. Use Query Designer to define your query. See ["Overview of creating a query in the Design tab" on page 74](#).
5. Use the default launch configuration to run and test your query application project. ["Default launch configuration" on page 98](#).
6. Use Scenario Browser to create and edit instances of queries that use parameters. See ["Using the Scenario Browser view" on page 111](#).
7. Iteratively update and test your query application as needed.
8. Export your project to an Ant deployment script. See ["Exporting to a deployment script" on page 60](#).

## Overview of creating a query in the Design tab

After you add a query file to a project, the Query Designer appears. You can define the query in the **Design** tab.

**Note:** By default, the query designer notifies errors at table level for Inputs and Pattern sections before entering data. After entering data, if there are errors in

any section, the errors are notified at the table level as well as row level of the table where there is an error.

Before you can define a query, you must define the events that you want the query to process. For information on how to write the event definitions, see "Overview of query application components" in *Developing Apama Applications*.

### To defining a query in the **Design** tab of Query Designer

1. In the **Inputs** pane, click the plus sign to add an event type definition that you want the query to process. You must add one or more inputs before you can create the event pattern you are interested in. See ["Adding query inputs " on page 77](#).
2. Optionally, in the **Parameters** pane, click the plus sign to add a query parameter. See ["Adding query parameters " on page 81](#).
3. From the event **Palette**, select an input event and drag it onto the pattern canvas. See ["Adding query patterns" on page 80](#).
4. In the **Actions** pane, click the plus sign to define an action that the query executes when it finds the specified event pattern. See ["Adding query actions " on page 88](#).
5. Optionally, In the **Conditions** pane, click the plus sign to add a filter, time constraint or exclusion. Conditions let you identify the exact events you are interested in. You must add one or more inputs before you can add query conditions. See ["Adding query conditions " on page 82](#).
6. Optionally, in the **Aggregates** pane, click the plus sign to add an aggregate expression to the find pattern. An aggregate expression aggregates event field values in order to find data based on many sets of events. See ["Adding query aggregates" on page 86](#).

At any time, you can click the **Source** tab to view the code that defines your query.

## Creating queries in Query Designer

As mentioned earlier, the Query Designer editor provides two views of a query definition:

- **Design** — Business analysts define and update query definitions in this view. The **Design** tab provides an event palette and a canvas for defining an event pattern of interest. Toolbars, tooltips, and dialogs make it easy to define the events of interest to the query as well as any required parameters, conditions for finding matches, aggregate calculations and actions to take when a match set is found.
- **Source** — Application developers can examine and update query definition source code in the EPL code editor available from the **Source** tab.

Query Designer keeps the two views synchronized. Any changes made in the **Design** view are reflected in the **Source** view and vice versa.

The following topics provide information and instructions for using Query Designer to create a query definition.

## Configuring a query


In the **Configure Query** dialog, you can configure the query to import event types and aggregate functions, query name and package, and metadata information.

Importing event types and aggregate functions lets you use the short name of an event type or an aggregate function instead of the full name. For example, when you are writing an aggregate:

- If you have imported the built-in aggregate function `avg ()`, to calculate the average value 'x' for all events 'e', you can write `avg (e.x)` instead of the fully qualified package name `com.apama.aggregates.avg (e.x)`.
- If you have not imported the built-in aggregate function `avg ()`, to calculate the average value 'x' for all events 'e', you have to write the fully qualified package name `com.apama.aggregates.avg (e.x)`.

**Note:** You can use the Configure Query dialog to clear the elements that are missing from the package.

### To configure a query




1. In the **Inputs** toolbar, click the **Configure Query** icon . The **Configure Query** dialog appears with the list of event types and aggregate functions.
2. Optionally, in the **Import definitions** tab, you can filter the event types and aggregate functions in the list using the drop down list at the text box.
  - Select **Show Events** to view only the event types.
  - Select **Show Aggregates** to view only the aggregates.
  - Select **Show Both** to view all the event types and aggregates.

3. In the **Import definitions** tab, select the required event type or the aggregate function.

In the text box, you can also type the name of the event type or the aggregate function that you want the query to import.

**Note:** You cannot select the event types or aggregate functions that are selected by default. These event types and aggregates are implicitly available as they share the same package as the query file.

4. In the **Name & Package** tab, you can edit the query name and package for an existing query file. If you do not specify a package in the **Package (optional):** field, then the query file is in the default package which contains EPL files that do not specify a package name.
5. In the **Metadata** tab, you can add, edit, or delete query metadata.

- To add a metadata entry, click . The Add metadata dialog appears. Enter the **Key** and **Value**, and click **OK**.
- To edit a metadata entry, select the entry you want to edit and click . The Edit metadata dialog appears. Edit the **Key** and **Value**, and click **OK**.
- To delete a metadata entry, select the entry you want to delete and click .

For more information on defining metadata, see "Defining metadata in a query", in *Developing Apama Applications*.


6. Click **OK**.

Optionally, when an event has been renamed or removed from the event definition file, then the missing events or aggregates appear in the **Configure Query** dialog. You can clear the missing events or aggregates in the **Configure Query** dialog by clicking **Clear List**.

## Adding query inputs

To specify the event pattern that you want your query to find, you must first define the event types that your query processes. The input definitions identify the events that you want the query to operate on. Perform these tasks in the **Inputs** pane in the **Design** tab of the Query Designer.

### To add a query input

1. In the **Inputs** toolbar, click the plus sign  to select an event type. The **New Query Input** dialog appears.
2. In the **Choose Event Input**, click the **Choose** button to select the event type you want to add in the **Event Type Selection**. Click **OK**.
3. In the **Window** tab:
  - **Event Window**
    - i. In the **Within** field, optionally specify a `float` expression or time literal that specifies the duration of time that an event remains as a current event that the query operates on. The query operates on the events that have been received within the time specified.  
  
For example, **Within: 7 days** implies that an event remains in the event window for 7 days.
    - ii. In the **Retain** field, optionally specify an `integer` expression that specifies the number of events to be retained in the window.  
  
For example, **Retain: 100** implies that a maximum of 100 events can be retained in the event window. If the number of events exceed 100, the older events are replaced by new events.

**Note:** You must specify a value in either the **Within** field or in the **Retain** field or both.

#### ■ **Partition**

In the **Key** field, optionally you can select one or more fields to be the query key. A query key identifies one or more fields in the events being operated on. Each input definition for a given query must specify the same key. Apama uses the key to partition the incoming events. A partition contains a set of events that have the same key value. The query processes the set of events in a given partition independently of every other partition. All the input events must have the same key names. If an event field does not have the same name as an existing input event, alias the key to the same name as the existing input key. For example, you can partition `Withdrawal` events according to their `account` numbers.

To arrange the keys in a required order, you must select the key and use the up and down arrows to arrange accordingly.

#### 4. In the **Filtering** tab:

##### ■ **Pre-Filtering (optional)**

In the **Filter** field, optionally specify an expression that filters the events you want the query to process. In the filter expression, you can specify event fields of type `boolean`, `decimal`, `float`, `integer`, `string` or `location`.

For example, if you want to filter `Withdrawal` events whose `amount` field is greater than 10, type `amount > 10.0`.

If you want to refer to a parameter in a filter, then you must first add the parameter. See ["Adding query parameters" on page 81](#).

##### ■ **Discard repeating values (optional)**

If there are multiple events with the same value for a field, the **With unique** functionality retains the latest value and discards the previous events with the same value.

- Select **Do not discard repeating values** if you want to retain events with repeating values.
- Select **Discard repeating values ("with unique")** if you want to remove events with repeating values.

For more information, see "Matching only the latest event for a given field", in *Developing Apama Applications*.

#### 5. In the **Timestamps (optional)** tab:

**Source Timestamps** — time at which an event happened at the event source that is available in the event. Use the **Timestamps (optional)** tab to configure the query to use **Source Timestamps**. The **Timestamps (optional)** tab controls whether the query runtime processes the events as they occur or whether the events contain some information

about the time they occurred. This process of Timestamps tab requires you to specify the **Wait**, **Time from**, and **Heartbeat** fields.

To configure the query to use source timestamps

- Select **No Timestamps - Process events at arrival time** to process the events as they arrive. You cannot specify the **Wait**, **Time from**, and **Heartbeat** fields if you select this option.
- Select **Use Timestamps - Wait for delayed events** to process the events according to the timestamps of the events.

**Note:** If events are delayed or re-ordered between the event source and the queries, then using the source timestamps where available yields more accurate results.

- **Event definition settings** — select this option if you want the query to use the Time from and Heartbeat settings provided by the event definition through annotations. If the annotation has a `DefaultWait` annotation, then the **Wait** field is initially set by copying the value from the `DefaultWait` annotation (if defined). But, you can still configure the **Wait** field.

Specify a `float` expression or a time literal in the **Wait** field that specifies the maximum delay for an event. For a query that uses source timestamps, you must specify the maximum time for the queries runtime to wait for an event before it can process. If the event does not define a `TimeFrom` annotation, use **Custom settings**.

- **Custom settings** — select this option if you have different queries using different settings or if there are no annotations. This ensures that not all queries are using the same Time from or Heartbeat. You can specify the **Wait**, **Time from**, and **Heartbeat** fields after selecting this option.
  - **Wait** — specify a `float` expression or a time literal that specifies the maximum time for the queries runtime to wait for an event before it can process.
  - **Time from** — select an action that returns source timestamps of the event to the query. The **Time from** drop-down lists actions that are legal and can be used time-from actions. The actions must not have any parameters and must return a float value.
  - **Heartbeat (optional)** — select the check box **Use heartbeat event (optional)** if you wish to configure an heartbeat settings. Click **Choose...** to select the **Heartbeat event type**. Select the event type and click **OK**. The selected event type must have the same key fields and time-from actions as the input event type. The heartbeat events with timestamps signal that the communication from the source to the queries system is working correctly.

**Note:** The heartbeat events can be used only if a query is using source timestamps.

For more information, see "Using source timestamps of events", in *Developing Apama Applications*.

6. Click **OK** to add this input definition to your query.

Query Designer displays the new input definition in a row in the **Inputs** section and the event type you specified appears in the **Palette**.

Repeat these steps for each event type you want your query to process. Apama recommends no more than four input definitions in a query. You cannot add the same event type as a query input more than once. The input event type must be unique.

For more information, see "Defining query input", in *Developing Apama Applications*.

## Adding query patterns

Before you can add a pattern to a query, you must add input definitions to the query. See ["Adding query inputs " on page 77](#).

---

### To add a query pattern to a query

1. If the query definition is not already displayed in Query Designer, expand the project's `queries` folder and double click the query file you want to edit.
2. In the Query Designer **Design** tab, from the **Palette**, drag and drop an event onto the canvas to the right.
3. To add a second event to the pattern, drag the event type from the **Palette** onto the canvas so that it hovers over the event that is already there.

The same event type can appear multiple times in a pattern. Query Designer automatically gives each event in the pattern a unique identifier.

Query Designer displays the followed-by ( $\rightarrow$ ) and `and` operators before and after the event that is already in the pattern.

4. Drag the event you are adding to the appropriate operator so that it is highlighted and drop it.

Query Designer displays the updated pattern.

5. To add another event to the pattern, drag it from the **Palette** onto the canvas.
6. To relate the event being added to one of the events already in the pattern, hover the event being added over the event that is already in the pattern.


To relate the event being added to two or more events already in the pattern, hover the event being added over the space between those events.

Query Designer displays the followed-by ( $\rightarrow$ ) and `and` operators before and after the event(s) that is (are) already in the pattern.

7. Drag the event you are adding to the appropriate operator so that it is highlighted and drop it.



Query Designer displays the updated pattern.

8. Repeat the previous three steps to add each additional event in the pattern you are looking for.
9. To add a period of time that must elapse before or after the event pattern occurs, drag **Wait** (  ) from the palette onto the canvas and hover over the pattern for prompts to indicate where you can drop it.

A waiting period is allowed at the beginning or end of an event pattern; it is not allowed between events.

10. Drag the **Wait** operator to the followed-by (->) operator at the beginning of the pattern or at the end of the pattern so that -> is highlighted and drop it.

Query Designer displays the updated pattern. The **Wait** operator has a blue color in contrast to the green color of the events. Inside the **Wait** operator, Query Designer assigns a unique identifier and default time period of 10 seconds.

11. To change the length of time, double click 10.0 in the Wait operator and enter the required time period according to ["Specifying a time period in Query Designer" on page 92](#).
12. If you need to delete an event or **Wait** operator from a pattern, right-click it and select **Delete**.
13. If you want to change the name of an event identifier in the pattern, double click that identifier and enter a new one.

The identifier you enter can contain uppercase letters (A-Z), lowercase letters (a-z), digits (0-9), underscores, pound signs (#), and dollar signs. Characters not allowed include spaces, digit as first character, any other special characters, and EPL keywords.

After you define the query's event pattern, you might want to add conditions that determine when the query successfully finds a match for the pattern. See ["Adding query conditions " on page 82](#).

## Adding query parameters

A query can specify one or more parameters. Parameters enable a query definition to function as a template for multiple query instances. A query that defines parameters is referred to as a parameterized query. An instance of a parameterized query is referred to as a parameterization.

A parameterized query offers the following benefits:

- Patterns of interest may be customized from a single generic query. This can significantly reduce the amount of code that needs to be written and maintained.
- Parameterizations exist only at runtime. There is no need to maintain a file for each instance.


- Parameters can be used throughout the query in which they are defined. For example, you can use them in the definition of inputs, in `find` actions, and in user-defined actions. Values do not need to be hardcoded.


You can add zero, one, or more parameters to a query.

For more information about how the use of parameters affect queries, see *Query lifetime in Developing Apama Applications*.

---

### To add a parameter to a query


1. In the Query Designer **Parameters** pane, click the plus sign  to add a new parameter.
2. In the **Parameters** pane, in the **Name** field, enter an identifier for the parameter.  
An identifier can include lowercase a-z, uppercase A-Z, digits, dollar sign, and underscore. An identifier cannot start with a digit and cannot include any other special characters.
3. Select the type of the parameter. A drop down in the **Parameters** pane provides the list of valid types applicable.

Query Designer displays the new parameter in a row in the **Parameters** pane. To modify the parameter, select it and then click the field to edit in the **Parameters** pane. To remove the parameter, select it and then click the **Delete** icon . Also, the new parameter is immediately added to the source code in the `parameters` section of the query definition. If you modify the parameter definition in the **Source** tab this is reflected in the **Parameters** pane in the **Design** tab.

## Adding query conditions

An Apama query can specify conditions that must be satisfied for there to be a match for the event pattern of interest. You can specify the following kinds of conditions:

- **Filter** — This is a Boolean expression that refers to the events in the specified pattern of interest. This expression must evaluate to true for there to be a match. There can be zero, one, or more query filter conditions. Adding a query filter condition inserts a `where` clause in the source view of the query.
- **Time constraint** — This is a time period during which some or all events in the pattern must have been received. There can be zero, one, or more time constraints. Adding a time constraint inserts a `within` clause in the source view of the query.
- **Exclusion** — This is the specification of an event whose presence prevents a match. If you need to prevent a match only when instances of this event have certain values you can add a filter that qualifies which instances of the specified event prevent a match. You can add zero, one or more exclusions. Adding an exclusion inserts a `without` clause in the source view of the query.

You add query conditions in the **Conditions** pane of the **Design** view. Click the down arrow to the right of the plus sign  and select the kind of condition you want to add. Query Designer displays a dialog in which you define the condition.

After you add a condition in the **Design** view, it appears in a row in the **Conditions** pane that shows the condition type, the expression that specifies the condition, and the range that the condition applies to if one was specified. Select the condition to edit or delete it.


When you add a condition it is immediately added to the source code. Any changes you make in the **Source** tab will be shown in the **Design** tab, and vice versa. If you introduce any errors into the query source code then you must resolve them in the **Source** tab before you can view the query in the **Design** tab.

For more details about the EPL code for query conditions, see "Query conditions" in *Developing Apama Applications*.

## Adding query condition filters (where)

A query can specify a Boolean expression that must evaluate to true for the query to find a match. This is referred to as a query condition filter. You can add zero, one, or more condition filters to a query.



### To add a condition filter to a query

1. In Query Designer, add the event inputs to the query and define the event pattern you want to find. See ["Adding query inputs" on page 77](#) and ["Adding query patterns" on page 80](#).
2. In the Query Designer **Conditions** pane, click the down arrow next to the plus sign  and select **Filter (where)**.
3. In the **New Query Condition Filter (where)** dialog, enter a Boolean expression that refers to the events in the pattern of interest.

For example, consider a `Withdrawal` event followed by another `Withdrawal` event, both for the same account. You are interested in this pattern only when the `country` fields in the two events are different. Using the aliases `w1` and `w2` for the two `Withdrawal` events, the following Boolean expression specifies the filter:

```
w2.country != w1.country.
```

4. Click **OK**.


The Query Designer displays the new filter in a row in the **Conditions** pane. To modify the filter, select it and click the **Edit** . To remove the filter, select it and click **Delete** . Also, the new filter is immediately added to the source code as a `where` clause in the `find` statement. If you modify the filter in the **Source** tab this is reflected in the **Conditions** pane in the **Design** tab.

## Adding query time constraints (within)

A query time constraint sets the time period during which the set of events that match the pattern of interest must have been received. A query can specify zero, one or more

time constraints. A time constraint can apply to the whole pattern or a portion of the pattern.

### To add a time constraint to a query



1. In Query Designer, add the event inputs to the query and define the event pattern you want to find. See ["Adding query inputs " on page 77](#) and ["Adding query patterns" on page 80](#).
2. In the Query Designer **Conditions** pane, click the down arrow next to the plus sign  and select **Time constraint (within)**.
3. In the **New Time Constraint** dialog, if the time constraint applies to all events in the pattern then leave **All events in the pattern** selected and skip the next step. Otherwise, select **Only specific events**.

For example, suppose you want to flag consecutive `Withdrawal` events for the same account that have different `country` field values. However, this is of interest only when the consecutive events both arrive within an hour. In this case, the time constraint applies to all events in the pattern.

4. To apply the time constraint to a portion of the pattern, select at least two events. The time constraint applies to only the events you select.

For example, suppose that the pattern of interest is an `A` event followed by a `B` event followed by a `C` event. However, this pattern is of interest only when the `C` event is received within 10 seconds after the `B` event. You want the time constraint to apply only to the `B` and `C` events and so you need to select those two events.

5. In the field at the bottom of the dialog, specify the period of time during which the selected events must be received. The elapsed time from when the first selected event is received to when the last selected event is received must be less than the time period you specify. See ["Specifying a time period in Query Designer" on page 92](#).
6. Click **OK**.


Query Designer displays the new time constraint in a row in the **Conditions** pane. To modify this time constraint, select it and then click the **Edit**  icon. To remove this time constraint, select it and then click the **Delete** icon . Also, the new time constraint is immediately added to the source code as a `within` clause in the `find` statement. If you modify the time constraint in the **Source** tab this is reflected in the **Conditions** pane in the **Design** tab.

### Adding query exclusions (without)

A query exclusion specifies that the presence of a particular event prevents a match. A query can specify zero, one or more exclusions. An exclusion can include a filter and/or a range to limit which instances of the identified event prevent a match.

**Note:** The information on the options in the **New Query Exclusion (without)** dialog which is provided below also applies when you modify an existing exclusion in the **Edit Query Exclusion (without)** dialog.

### To add an exclusion condition to a query

1. In Query Designer, add the event inputs to the query and define the event pattern you want to find. See ["Adding query inputs " on page 77](#) and ["Adding query patterns" on page 80](#).
2. In the Query Designer **Conditions** pane, click the down arrow next to the plus sign  and select **Exclusion (without)**.
3. In the **New Query Exclusion** dialog, in the **Event Type** field, select the event whose presence prevents a match. You can select only an event type that has been added as a query input. If the type you want to select is not listed then you must first add it as a query input.

For example, consider consecutive `Withdrawal` events that have different `country` field values. You want an alert when this occurs except when a `TravelNotification` event is received before the second `Withdrawal` event. In this case, select `TravelNotification` as the event type that prevents a match.

4. In the **Name** field, Query Designer displays a name that it automatically generates. This is an alias for the selected event and is unique across all aliases in the query. You have the option of using this alias in the next step.
5. In the field in the middle of the dialog, you can optionally specify a Boolean expression that limits which event instances prevent a match. In this Boolean expression, you can optionally refer to the alias that appears in the **Name** field. When the specified Boolean expression evaluates to true the presence of the specified event prevents a match. When this expression evaluates to false the presence of the specified event allows a match.


For example, you might want a `TravelNotification` event to prevent a match only when its `date` field has the same value as the `date` field of the first `Withdrawal` event. If the event aliases `w1`, `w2`, and `t1` are used, the following Boolean expression defines the exclusion filter: `w1.date = t1.date`. When the `date` fields of the first `Withdrawal` event and the `TravelNotification` event have the same value then the presence of the `TravelNotification` event between two consecutive `Withdrawal` events will prevent a match and an alert will not be sent. When the `date` fields of the first `Withdrawal` event and the `TravelNotification` event have different values then the presence of the `TravelNotification` event between two consecutive `Withdrawal` events will allow a match and an alert will be sent.

6. At the bottom of the dialog, indicate when the presence of the selected event prevents a match. If you accept **During the whole pattern**, then you can select **OK** and you are done adding an exclusion. If the presence of the selected event prevents a match only when it is received **In between specific events**, then select that option and then select at least two events.
7. Click **OK** to save the query exclusion.

Query Designer displays the new exclusion in a row in the **Conditions** pane. Also, the new exclusion is immediately added to the source code as a `without` clause in the `find`

statement. You can click the **Source** tab to view the exclusion in the source code. Any changes you make in the **Design** tab are reflected in the **Source** tab, and vice versa.


To modify the exclusion, select it in the **Conditions** pane and then click the **Edit**  icon.

To remove the exclusion, select it in the **Conditions** pane and then click the **Delete**  icon.

## Adding query aggregates

An Apama query can specify aggregates that let the query find data based on many sets of events. You can add one or more of each of the following to your query definition:

- **Aggregate calculation** — Adding an aggregate calculation inserts a `select` clause in the source view of the query.
- **Aggregate filter** — This is a Boolean expression that refers to the events in the specified pattern of interest. This expression must evaluate to true for there to be a match. There can be zero, one, or more query filter conditions. Adding an aggregate filter inserts a `having` clause in the source view of the query.

You add query aggregates in the **Aggregates** pane of the **Design** view. Click the down arrow to the right of the plus sign , and select the kind of aggregate you want to add. Query Designer displays a dialog in which you define the aggregate. The following topics provide more information:

- ["Adding query aggregate calculations" on page 86](#)
- ["Adding query aggregate filters" on page 87](#)

After you add an aggregate in the **Design** view, it appears in a row in the **Aggregates** pane that shows the aggregate type, the expression that specifies the aggregate, and an alias for the aggregation. Select the aggregate to edit or delete it.

When you add an aggregate, it is immediately added to the source code. Any changes you make in the **Source** tab will be shown in the **Design** tab. If you introduce any errors into the query source code then you must resolve them in the **Source** tab before you can view the query in the **Design** tab.

For more details about the EPL code for query conditions, see "Aggregating event field values" in *Developing Apama Applications*.


## Adding query aggregate calculations

A query can specify an aggregate calculation that computes a single value based on multiple events, such as the average of a series of numbers. For an overview of all built-in aggregate functions, see "Built-in aggregate functions" in *Developing Apama Applications*.

When a query does not specify an aggregate calculation or filter, only the most recent set of events that match the pattern are used to invoke the actions and any other procedural code. With an aggregate calculation or filter, every set of events, in the current set of

events, that matches the pattern is available for use by the aggregate function, provided that the latest event is present in one of the sets of events. Any events or combinations of events that do not match the pattern or do not meet specified query conditions, are ignored; their values are not used by the aggregate function.

### To add an aggregate calculation to a query

1. In the Query Designer **Aggregates** pane, click the down arrow next to the plus sign  and select **Calculation (select)**.

2. In the **New Query Aggregate Calculation** dialog, specify an expression that uses an aggregate function with identifiers from the pattern. You can also use parameters.


The `first()` and `last()` built-in aggregate functions process the oldest or newest event in the current set of events that the query is processing, respectively.


For example, the following aggregate calculation determines the average price of trades:

```
avg(trade1.price, trade1.amount)
```

3. The field at the bottom of the dialog displays the identifier for this calculation. You can change the identifier. You can refer to this identifier in a query action.
4. Click **OK**.

Query Designer displays the new aggregate calculation in a row in the **Aggregates** pane.

To modify the aggregate calculation, select it and then click the **Edit**  icon.

To remove the calculation, select it and then click the **Delete** icon .

Also, the new calculation is immediately added to the source code as a `select` clause in the `find` statement. If you modify the calculation in the **Source** tab, this is reflected in the **Aggregates** pane in the **Design** tab.

You can add one or more aggregate calculations as well as one or more aggregate filters. The order in which they appear in the **Aggregates** pane is the order in which they are executed in the query. To change the order, select an aggregate and click the up arrow or down arrow as needed.

When you add more than one aggregate filter to a query, then all of them must evaluate to true for the query actions to be invoked.

See also "Aggregating event field values" in *Developing Apama Applications*.

## Adding query aggregate filters


A query can specify a Boolean expression that uses an aggregate function that must evaluate to true for the query actions and any other procedural code are invoked. For an overview of all built-in aggregate functions, see "Built-in aggregate functions" in *Developing Apama Applications*.

When a query does not specify an aggregate calculation or filter, only the most recent set of events that match the pattern are used to invoke the actions and any other procedural



code. With an aggregate calculation or filter, every set of events, in the current set of events, that matches the pattern is available for use by the aggregate function, provided that the latest event is present in one of the sets of events. Any events or combinations of events that do not match the pattern or do not meet specified query conditions, are ignored; their values are not used by the aggregate function.

### To add an aggregate filter to a query

1. In the Query Designer **Aggregates** pane, click the down arrow next to the plus sign  and select **Filter (having)**.
2. In the **New Query Aggregate Filter** dialog, specify a Boolean expression that uses an aggregate function with identifiers from the pattern. You can also use parameters.

The `first()` and `last()` built-in aggregate functions process the oldest or newest event in the current set of events that the query is processing, respectively.

For example, this aggregate filter causes the query actions to be invoked only when the average price of trades is greater than 100.0:

```
avg(trade1.price) > 100.0
```

3. Click **OK**.

Query Designer displays the new aggregate filter in a row in the **Aggregates** pane.

To modify the aggregate filter, select it and then click the **Edit**  icon.

To remove the filter, select it and then click the **Delete** icon .

Also, the new filter is immediately added to the source code as a `having` clause in the `find` statement. If you modify the filter in the **Source** tab this is reflected in the **Aggregates** pane in the **Design** tab.

You can add one or more aggregate calculations as well as one or more aggregate filters. The order in which they appear in the **Aggregates** pane is the order in which they are executed in the query. To change the order, select an aggregate and click the up arrow or down arrow as needed.

See also: "Aggregating event field values" in *Developing Apama Applications*.


## Adding query actions

In a query definition, you specify the actions you want Apama to perform when it finds a match for the event pattern specified in the query. You can add one or more actions and there are two kinds of actions that you can add:

- Send event actions send an event to a channel that you specify.
- Custom EPL actions can do anything that can be defined in an EPL action.

Before you define query actions, you should add the query input events and specify the query pattern. If this is a parameterized query, you should add the query parameters before you add actions. You can then refer to the parameters in the actions.



You add query actions in the **Actions** pane of the **Design** view. Click the down arrow to the right of the plus sign  and select the kind of action you want to add. Query Designer displays a dialog in which you define the action. See one of the following topics for details:

- ["Adding query send event actions" on page 89](#)
- ["Adding query custom EPL actions " on page 91](#)

After you add an action in the **Design** view, its name appears in a row in the **Actions** pane. Select the action name to edit or delete it.

When you add an action, it is immediately added to the source code. Any changes you make in the **Source** tab will be shown in the **Design** tab, and vice versa. If you introduce any errors into the query source code then you must resolve them in the **Source** tab before you can view the query in the **Design** tab.


For more details about the EPL code for query conditions, see "Query conditions" and "Specifying actions in event definitions" in *Developing Apama Applications*.

## Adding query send event actions

In the Query Designer **Actions** pane, add the send event action(s) you want Apama to perform when it finds a match for the event pattern specified in the query. You can specify one or more send actions as well as one or more custom actions.

Before you define query actions, you should add the query input events and specify the query pattern. If this query uses parameters, you should add the query parameters before you add actions. You can then refer to the parameters in the actions.


To add a send action to a query

1. In the **Actions** pane of the Query Designer, click the drop down arrow to the right of the plus sign  and select **Send Event** to create a send event action. The **New Query Send Event Action** dialog appears.
2. Click **Choose**, and select the **Event** type.
3. In the **Name** field, enter a name for the send event action.
4. In the **Description** field, optionally enter a description of what the action performs.
5. In the **Channel** field, specify an EPL expression to which the event will be sent. The EPL expression must evaluate to a string. If you are using a string literal, you have to enclose the string literal within quotes.

For example, in the sample `Delayed_InOrder_Events_Car_Sample`, you can use an EPL expression `caralert.carId` in the **Channel** field for the event type `apamax.querysamples.delayedcarevents.CarEngineAlert`. Here, "caralert." is a channel name prefix which is combined with the `carId` value. If the value of `carId` is 10, then the channel to which the event will be sent is `caralert.10`.

- To insert an event field or an expression, right-click the **Channel** field and select the required event field or expression.

- To convert an expression to a plain text:
    - i. Select the expression that you want to convert to plain text
    - ii. Right-click the selected expression
    - iii. Select **switch to plain text [ctrl-p]**
  - To convert plain text to an expression:
    - i. Select the plain text that you want to convert to an expression
    - ii. Right-click the selected plain text
    - iii. Select **switch to expression [ctrl-e]**
6. In the **Fields** table, specify a value to every field name.

To specify a value to a field name, select the field name and click the edit sign  .  
The Edit dialog appears.

In the Edit dialog:

- To insert an expression, right-click in the text box and select the required expression. Click **OK**.
- To convert an expression to a plain text:
  - i. Select the expression that you want to convert to plain text
  - ii. Right-click the selected expression
  - iii. Select **switch to plain text [ctrl-p]**
  - iv. Click **OK**
- To convert plain text to an expression:
  - i. Select the plain text that you want to convert to an expression
  - ii. Right-click the selected plain text
  - iii. Select **switch to expression [ctrl-e]**
  - iv. Click **OK**

The **Fields** table displays event fields and the values to be specified for the event. Values can be any valid EPL expression. Only string literals need to be enclosed within quotes. If any fields that are not present in the event definition have been incorrectly added to the Send Event action's event in the **Source** editor tab, then these fields are listed in the missing fields table. These missing fields and values are persisted with the **Send Event** action when you click **OK**. If you do not want these missing fields and values to persist, click **Clear List**. **Clear List** removes all the missing fields and values from the **Send Event** action.

7. Click **OK** when the send action definition is complete.

If you look at the **Source** view of the query, you can see that for each action you define, Query Designer inserts a `%send` statement in the EPL query definition. The `%send()` statements appear in the `find` statement block and can be interleaved with regular EPL.

In the **Source** view, the modifications you can make to `%send` statements are to modify the quoted name, the quoted description, and the channel. Any changes you make in the **Source** view will appear in Query Designer. If there is a syntax error in the **Source** view, you cannot display the **Design** view until you correct the error. The recommendation is that only Query Designer should insert `%send()` statements. At runtime when the query finds a match, it executes all specified actions in the order in which they are listed in the **Actions** pane.

To edit a send event action, select the send event action and click the edit sign .

To delete a send event action, select the send event action and click the delete sign .

The arrows at the top of the **Actions** pane let you change this order. To change the order, select the action you want to move and then click the appropriate arrow as many times as needed.


## Adding query custom EPL actions

In the Query Designer **Actions** pane, add the custom EPL actions you want Apama to perform when it finds a match for the event pattern specified in the query. You can specify one or more custom actions as well as one or more send actions. See ["Adding query actions" on page 88](#) for an understanding of these two kinds of actions.

Before you define query actions, you should add the query input events and specify the query pattern. If this is a parameterized query, you should add the query parameters before you add actions. You can then refer to the parameters in the actions.

---

### To add a custom action to a query

1. In the Query Designer **Actions** pane, click the down arrow to the right of the plus sign  and select **Custom EPL** to display the **New Query Custom EPL Action** dialog.
2. In the **Name** field, enter a name for the custom action.
3. In the **Description** field, optionally enter a description of what the actions does.
4. In the **Custom EPL code editor**, enter the EPL code that defines the action. You define an action in a query in the same way that you define an action in a monitor except that when you use the **New Query Custom EPL Action** wizard, you cannot specify action arguments nor an action return value. The numbers in the left column approximately indicate the lines that the code you enter will be on if you click the **Source** tab to view the query code.

As you define your action, Query Designer can help you with completion proposals. See ["Obtaining content assistance" on page 47](#) and ["Using auto-completion" on page 47](#).

5. The **Errors** field displays information about any errors in the EPL you enter. You must resolve any errors before you can save the action.

6. Click **OK** when the action definition is complete.

If you look at the **Source** view of the query, you can see that for each action you define, Query Designer inserts a `%custom` statement in the EPL query definition. The `%custom()` statements appear in the `find` statement block and can be interleaved with regular EPL.

In the **Source** view, the modifications you can make to `%custom` statements are to modify the quoted name, the quoted description, or the EPL in the block. The block contains the action definition you entered in the **Custom EPL code editor**. Any changes you make in the **Source** view will appear in Query Designer. If you make an invalid change in the source then you cannot display the **Design** view until you correct the error.

The recommendation is that only the Query Designer should insert `%custom()` statements.

7. At runtime, when the query finds a match it executes all specified actions in the order in which they are listed in the **Actions** pane. Two arrows at the top of the **Actions** pane let you change this order. To change the order, select the action you want to move and then click the appropriate arrow as many times as needed.

## Specifying a time period in Query Designer

In Query Designer, you can specify a length of time:

- To wait before or after an event pattern occurs. This is in the **Wait** operator in an event pattern.
- In which all events in the pattern or some events in the pattern must be received. This is in a query time constraint condition.

When you specify a time period, it must be one of the following:

- A time literal, which is an `integer` or `float` value, followed by a space, followed by one of the following:
  - `day`
  - `days`
  - `hour`
  - `hours`
  - `min`
  - `minute`
  - `minutes`
  - `sec`
  - `second`

- seconds
- msec
- millisecond
- milliseconds
- Insert a space between consecutive time literals. For example, you could enter 5 min 10.0 sec.
- A float literal, which can be a float value or an expression that resolves to a float value. An expression can use constants and parameters. A float literal always indicates a number of seconds.

For example, the following specifications are valid:

- 10 sec
- 3.5 hours
- 5 days
- 90 min 30.0 sec
- 60.0

## Editing source code in Query Designer

The Query Designer provides a **Source** tab to view or modify the EPL source code for the query definition. Changes made in the **Design** tab are reflected in the **Source** tab, and vice versa.

Query Designer provides an **Actions** pane that lets you specify actions to perform when a match set is found. For each action added in the **Actions** pane, Query Designer inserts a `%custom` statement in the EPL query definition. The recommendation is that only Query Designer should insert `%custom()` statements.

The `%custom()` statements appear in the `find` statement block and can be interleaved with regular EPL. `%custom()` signifies a block of EPL that was added in the **Actions** pane of the **Design** tab. You can modify the string that contains the name of the action, the string that contains the description of the action, or the EPL inside the braces, `{` and `}`, of the `%custom()` statement. You should not modify other content inside the parentheses.

For example, you might see something like this in the code editor:

```
package com.apama.samples.usercomplaints;

query FindUsersComplainingAfterNoService {

    inputs {
        SupportCall() key subscriberId within 3 days;
        CallDataRecord() key customerId as subscriberId within 3 days;
    }


    // Identifies subscribers who have not been able to make any successful
    // calls after 2 days,
```

```
// and have made two support calls, the second of which escalated the
// problem.
find wait (2 days):previous -> SupportCall:firstCall
    -> SupportCall:secondCall
    where secondCall.escalated
    without CallDataRecord:cdr where cdr.callSucceeded between
    (previous firstCall)
    {
        %custom("title":"sendCustomerEscalationAlert","description":"Send
            a warning about escalated a support call")
        {
            send CustomerEscalationAlert(firstCall.subscriberId,
            "User escalated support request") to
            "com.apama.samples.user_complaints";
        }
    }
}
```

If you want to change the syntax coloring that is used for queries, see the description of the Apama preference page ["Syntax Coloring" on page 159](#).

## Errors in query definitions

In the **Design** tab of Query Designer, as you add and edit the information that defines the query, Query Designer immediately displays error messages if you enter text that is not allowed or if you leave out a required item. Query Designer is optimized to prevent you from saving a query definition that contains an error. However, if there is an error in the query definition then Software AG Designer displays an error icon in the Query Designer section that defines the part of the query that contains the error, for example:

 **Inputs** [Errors \(1 Item\)](#)

This appears in the toolbar of the section that contains the error. Hover over the "Errors (1 Item)" text, for example, to display a message that describes the error. You must resolve the error before Query Designer can save your query definition.

Any syntax errors in the **Source** tab prevents the **Design** tab from being used. You must correct any syntax errors in the **Source** tab before you can view the query in the **Design** tab..

A query file fails to run in the Correlator at the time of launching the Apama project if the query file has the following errors:

- Syntax error — These errors prevent the **Design** tab from appearing. The syntax errors include errors such as mismatching brackets or quotes, missing semicolon (;) from end of statements.
- Validation error — These errors prevent reporting of semantic errors, and are highlighted in the sections of the **Design** tab. The Completion proposals do not appear in the Custom EPL dialog. The validation errors include errors such as inconsistent keys across the inputs.
- Semantic error — These errors are highlighted in the sections of the **Design** tab if there are no validation errors. The semantic errors include errors such as using the wrong type in an expression, or using a variable or field that does not exist.

## Enabling diagnostic logging

You can enable diagnostic logging for a query or a scenario file in **Project Explorer** view (available in all perspectives) and **Minimal Navigator** view (available in **Apama Workbench**) perspective. If you enable this feature, the selected query or the scenario is generated with extra diagnostic logging when you run the project. A query with diagnostic logging enabled will have extra logging at both INFO and DEBUG level. To view the DEBUG level logging, you must run the correlator with its log level set to DEBUG. For more information, see ["Correlator arguments" on page 102](#).

For information on logging attributes, see "Setting logging attributes for packages, monitors and events" in *Deploying and Managing Apama Applications*.

---

### To enable diagnostic logging

1. In **Project Explorer**, right-click the query or the scenario file.
2. Click **Apama > Diagnostic Logging**.

A check mark appears for **Diagnostic Logging** specifying that diagnostic logging is set to true for the selected file.

## Configuring query projects

---

For a query application that will be deployed on a cluster of correlators, you need to add Java Message Service (JMS) support and Universal Messaging (UM) support to your project. See ["Correlator arguments" on page 102](#) and ["Adding Universal Messaging configuration to projects" on page 46](#).

Also, you must map JMS messages and EPL events. See *Mapping Apama events and JMS messages in Connecting Apama Applications to External Components*.

## Exporting query deployment scripts

---

To deploy an Apama query application, you can export the project's launch configuration to create a deployment script. This generates the build files, configuration files, property definition files, scripts, and EPL files and copies other resources such as dashboards that are used by Ant to build and launch the project on a different machine.

See ["Exporting to a deployment script" on page 60](#).





# 4 Launching Projects

---

■ Running Apama projects .....	98
■ Monitoring Apama applications .....	107

As you develop your applications, you can test them by running them from Software AG Designer. Projects contain *launch configurations* that specify which resources in the project need to be started and any initialization information they need. By default, a single launch configuration is created for each project the first time it is launched, but you can create others to test an application with a different set of monitors or a different version of a block.

For information about creating launch configurations for debugging and profiling purposes, see the following:


- ["Debugging EPL Applications" on page 115](#)
- ["Debugging JMon Applications" on page 121](#)
- ["Profiling EPL Applications" on page 129](#)


## Running Apama projects

Software AG Designer uses a project's launch configuration to start the application. You can use the project's default launch configuration or you can define one or more configurations for launching your Apama project in different ways.

### Default launch configuration

An application's default launch configuration starts the default correlator and uses all the monitors, events and queries defined in the project. A project's default launch configuration is available in both the Workbench and Developer perspectives.

To start an Apama project with the project's default configuration, click . This icon is enabled only if an Apama project is selected in the **Project Explorer**. When you run the Apama project, the project is started with default configuration. If the Apama project does not contain any configuration, a default configuration is applied before starting the project.




To stop an Apama project, click . This icon is enabled only if a running Apama application is selected in the **Project Explorer**.

**Note:** The icons are disabled if you select more than one project in the **Project Explorer**.

The Apama project must contain `.deploy` file to run the Apama project. If `.deploy` is missing, the Apama project fails to start with incomplete configuration and you have to reset the configuration. To reset the launch configuration, right-click the Apama project and select **Run as > Run Configurations....** The `.deploy` file is created.


## Workbench perspective

### To start the default launch configuration for the current project from the Workbench perspective

1. In the Workbench Project view, click the Start button . This starts the default correlator with all the project's monitors, events and scenarios. If the project includes a default dashboard, Software AG Designer launches it in the Dashboard Viewer. Information about the running application is shown in the Console view and if the project has a scenario, the Scenario Browser view appears.
2. To restart the application, click the **Restart** button  in the Workbench Project view.
3. To stop the click the **Stop** button  in the Workbench Project view. In addition to halting the application, this closes the **Dashboard Viewer**

## Developer perspective

### To start the default launch configuration for a project from the Developer perspective

1. In the Project Explorer view, select the project you want to run.
2. Select **Run > Run** from the menu. This starts the default correlator with all the project's monitors, events and scenarios. If the project includes a default dashboard, it is launched in Dashboard Viewer. Information about the running application is shown in the Console view.
3. If you want to inspect the behavior of the project's scenarios in the Scenario Browser, select **Window > Show View > Scenario Browser**.
4. To stop a running application click the **Terminate** button  in the Console view. If the launch configuration started Dashboard Viewer, you may want to stop Dashboard Viewer before stopping the application otherwise when you stop the application Dashboard Viewer will display a message its connection to the correlator was lost.

## Defining custom launch configurations

In many cases, you may want to create custom launch configurations, for example to run your applications with a subset of the monitors in your project or if your application relies on an IAF adapter. Note that the Apama Workbench perspective is targeted at creation of simple applications and so does not provide much support for projects with multiple launch configurations. If you need the power of multiple launch configurations, use the Apama Developer perspective.

When you create a new shared custom launch configuration, two files are created and are placed in the directory specified by **Shared file**. The names of these files have the following format and the launch configuration information is split between them:

■ `launch_config_name.deploy`

■ `launch_config_name.launch`

Any changes you make to the launch configuration will be reflected in the `.deploy` file. However, if you export a launch configuration to an Ant deployment script and then the shared location changes, then you must re-export the launch configuration to a new Ant deployment script. If you do not, the old Ant deployment script fails because it cannot find the `.deploy` file.

---

### To create a launch configuration

1. In the Project Explorer view, select the project you want.
2. Do one of the following:
  - If you are using the Apama Developer perspective, select **Run > Run Configurations** from the menu. The Create, manage, and run configurations wizard starts. The wizard has four tabs, the **Apama Project** tab, the **Components** tab, the **Environment** tab, and the **Common** tab.
  - If you are using the Apama Workbench perspective, select the project of interest in the Workbench Project view and then click the **Edit** button just above the **Start**, **Stop**, and **Restart** buttons in the launch control panel. This allows the default launch configuration for the project to be edited. Creating multiple launch configurations is not recommended for Workbench perspective users.
3. In the wizard's **Name** field, assign a name to the launch configuration.
4. On the **Apama Project** tab, fill in the following information:
  - **Project** — Specifies the project to launch.
  - **Dashboard** — Contains information about launching a dashboard when the project runs. It contains the following fields:
    - **Open Dashboard Viewer during launch** — Specifies whether or not the Dashboard Viewer should be launched when running the project. The default is to run the Dashboard Viewer.
    - **Use default dashboard** — The default dashboard is the dashboard project in the `dashboards` directory of the project. You can launch another dashboard by disabling the default dashboard and specifying the project relative path of the dashboard to launch.
    - **Viewer Arguments** — Specify any arguments to be added to the end of the command line for the dashboard viewer process.
5. The **Components** tab lists the components that are needed by the project such as additional correlators or external processes. You can add and remove components, edit their specifications, change the order in which they are started, or specify connections between correlators. Components are started in the order in which they appear in the **Components** tab from top to bottom. However, there is no waiting for one component to finish its startup before the next component is started. In other words, you cannot depend on startup for one component to already be complete when a subsequent component is started.

You can use the following buttons:

- **Up** — Moves the selected component up in the order in which it starts.
  - **Down** — Moves the selected component down in the order in which it starts.
  - **Edit** — Allows you to modify the settings for the selected component.
  - **Remove** — Removes the selected component from the launch configuration.
  - **Add** — Adds a correlator or an external process to the launch configuration. To add a component, click the **Add** button and select the type of component you want to add.
  - **Restore Default** — Sets the launch configuration to use the default launch configuration.
  - **Connections** — Displays the **Connections** dialog, which lets you add and remove connections between correlators. See ["Connecting correlators" on page 105](#).
6. The **Environment** tab lists any additional environment variables needed to run any of the processes started by this launch configuration.

You can use the following buttons:

- **New** — Specifies a new environment variable.
- **Select** — Selects an environment variable from the list of Eclipse environment variables.
- **Edit** — Modifies the value of an environment variable.
- **Remove** — Removes an environment variable from this configuration.

Apama recommends that you append new environment settings to the native environment as otherwise it will be necessary to manually specify the standard Apama environment variables in order for the process to start correctly.

Note that to add a suffix or prefix to an existing environment variable, a new environment variable of that name should be created, and the existing value specified as part of the variable's value, for example, `PATH=${env_var:PATH};C:\my path`.

7. The **Common** tab specifies additional attributes for running this launch configuration.

You can specify the following information:

- **Save as**
  - **Local file**
  - **Shared file** — This is the default selection and the default path is the `config\launch` folder in your project directory. Click **Browse** to navigate to and select another location that is available to all users sharing this project.

- **Display in favorites menu** — Select the appropriate checkbox to display this configuration as a choice in the drop-down menu of the **Debug** or **Run** button in the toolbar.
- **Console encoding** — The default encoding for output to the console is **Cp1252**. To encode console output in a different format, click **Other** and select the encoding you want. For example, **UTF-8**.
- **Standard Input and Output**
  - **Allocate console (necessary for input)** — For correlator launch configurations, there are no reasons not to allocate a console. Allocating a console does not affect performance.
  - **File** — If you want to capture correlator output in a file, navigate to and select a file to contain the correlator output.
  - **Append** — When you capture correlator output in a file, indicate whether you want to append the output to the specifies file.
- **Launch in background** — The default is that the correlator runs as a background process.

## Adding a correlator

When you add a correlator to the launch configuration, the **Correlator Configuration** dialog is shown, which includes these tabs:

- **Correlator Arguments**
- **Persistence Options**
- **Injects**
- **Event Files**

## Correlator arguments

The **Correlator Arguments** tab specifies the options and arguments that Software AG Designer uses to start the correlator.

You can accept the default values or change one or more arguments. You can also add arguments. In addition, you can also control the behavior with the following options:

- **Hostname** — `localhost`. The name of the correlator host machine.
- **Port** — This is the port on which the correlator listens for monitoring and management requests.
- **Log level** — Default is INFO.
- **Log to file** — Default is `${LAUNCH_CONFIG_NAME}_${CORRELATOR_NAME}.output.log`. This is the correlator status log.

- **Input log** — Default is `Correlator_1_${START_TIME}_${ID}.input.log`. This log contains all incoming messages. See "Using an input log to diagnose problems" in *Deploying and Managing Apama Applications*.
- **Xconfig** — Select this option to provide additional configurations to the correlator using a text file. For example, per-package logging settings. See "Using the Apama component extended configuration file" in *Deploying and Managing Apama Applications*.
- **Reuse Correlator** — Select this option to use an already running correlator.
- **Clear Correlator** — Select this option to delete all the correlator contents when the launch starts.
- **Java Support** — Select this option to provide Java support for JMon applications. Normally this is automatically selected when a JMon application is created in Software AG Designer.
- **UM Messaging** — Select this option to inject the project's `UM-config.properties` file, which is in the project's `config` folder. This is equivalent to specifying the `-UMconfig` option when starting a correlator. Uncheck this box if you want to run the project without using UM. If the project does not contain a UM configuration properties file, this option is disabled.  
  
See "Using Universal Messaging" in *Connecting Apama Applications to External Components*.
- **JMS Support** — Select this option to provide JMS support for correlator-integrated messaging. Normally this is automatically selected when the correlator-integrated adapter for JMS is added to an Apama application.
- **Distributed MemoryStore Support** — Select this option to provide support for distributed MemoryStore. With the distributed MemoryStore, you can create distributed stores that provide access to data that will be shared among Apama applications running in separate correlators.
- **Connectivity Plug-ins Support** — Select this option to provide support for Connectivity plug-ins that run inside the correlator process to allow messages to be sent and received to/from external systems.
- **Maximum java heap size in Mb (-Xmx)**. Default is 512.
- **Maximum java off-heap storage in Mb (-XX:MaxDirectMemorySize)**. Default is 16384.
- **Extra command line arguments**. Specify any additional arguments for starting the correlator. See "Starting the event correlator" in *Deploying and Managing Apama Applications*.

## Persistence options

The **Persistence Options** tab specifies the correlator persistence settings Software AG Designer will use when it runs this launch configuration.

- **Enable correlator persistence** — When set, this tells the correlator to start with the persistence option.
- **Directory location** — This specifies the location on disk of the persistent store used to preserve correlator state.
- **Startup options** — These options specify the startup behavior of the correlator with respect to how it handles its persistent store.
  - **Clear state and re-inject** — Specifies that you want to clear the contents of the recovery datastore.
  - **Resume from persistent state** — Specifies that the correlator will restart using the state from its persistent store as of the last committed snapshot.
  - **Prompt for startup options dialog during correlator launch** — Specifies that Software AG Designer will present a dialog when launching the application. The dialog asks whether to clear the correlator state information and re-inject the application code or to resume from the state information stored in the last committed snapshot.

For more information on correlator persistence, see "Using Correlator Persistence" in *Developing Apama Applications*.

For more information on persistence options for starting the correlator, see "Starting the event correlator" in *Deploying and Managing Apama Applications*.

## Injections

The **Injections** tab lists the files, except event files, that will be injected or sent into the correlator when Software AG Designer runs this launch configuration. This includes the files that are in your project as well as additional files that have been added to the project build path. See ["Defining custom launch configurations" on page 99](#).

By default the **Injections** tab lists all EPL files in the project. The files are listed in dependency order. This is the order in which Software AG Designer will load the files. Software AG Designer determines the dependency order when it builds the project. In almost all projects, you do not need to change the order. However, if necessary, you use this tab to change the order in which Software AG Designer injects the project's EPL files.

When **Automatic Ordering** is selected you have the option to exclude files from the launch configuration by unchecking them. When you run this project the unchecked files are not injected.

When you select **Manual Ordering** the **Up** and **Down** buttons appear. Select the file whose position you want to change and click the **Up** or **Down** button as many times as needed. When **Manual Ordering** is selected, Software AG Designer does not resolve any conflicts. It is up to you to correctly order the files.

If you uncheck a check box, the corresponding file remains in the list, but will not be loaded during this launch configuration.



All selected EPL files are injected and then all selected `.evt` files (on the **Event Files** tab) will be injected.

## Event Files

The **Event Files** tab lists the event files that will be injected or sent into the correlator when Software AG Designer runs this launch configuration. This includes event files that are in your project as well as additional event files that have been added to the project build path. See ["Defining custom launch configurations" on page 99](#).

By default the **Event Files** tab lists all `.evt` files in the project. The files are listed in lexicographic order by file name. This is the order in which the files will be injected.

If necessary, you use this tab to change the order in which the project's event files are to be injected. Select the file whose position you want to change and click the **Up** or **Down** button as many times as needed. It is up to you to ensure that the files are injected in the correct order.

If you uncheck a check box, the corresponding event file remains in the list, but will not be injected during this launch configuration.

All selected EPL files (listed on the **Injections** tab) are injected and then all selected `.evt` files will be injected.

## Connecting correlators

When you are creating a custom launch configuration you can specify connections between correlators. This is similar to connecting correlators with the `engine_connect` correlator utility.

Each connection you add goes in only one direction. For example, suppose you define a connection from `correlatorA` to `correlatorB`. This connection lets `correlatorA` send events to `correlatorB`. If you want `correlatorA` to receive events from `correlatorB` you must add a second connection that specifies `correlatorB` as the source of the connection and `correlatorA` as the target of the connection.

---

### To connect correlators

1. In the **Run Configurations** dialog, select the **Components** tab and ensure that the correlators you want to connect are listed as components. See ["Adding a correlator" on page 102](#).
2. Click **Connections**.
3. In the **Connections** dialog, click **Add**, which activates a row in the connections table.
4. In the activated row, in the **From** field, select the name of the correlator that is the source of this connection. If the name of the correlator you want to connect does not appear, click **Cancel** to return to the **Components** tab and then select **Add > Add Correlator**.
5. In the **To** field, select the name of the correlator that is the target of this connection.

6. In the **Channel** field, specify one or more channels to be used by this connection. Use a comma as a separator. For example, `channelA, channelB`.
7. In the **Flags** field, specify any options that are supported by the `engine_connect` utility.

See "Event correlator pipelining" in *Deploying and Managing Apama Applications*.

After adding the desired connections, click **OK** to return to the **Components** tab.

To remove a connection, select the row that defines the connection and click **Remove**.

## Adding an external process

When you add an external process to the launch configuration, the External Process Configuration dialog is shown.

This dialog allows the user to add and edit an external process associated with the launcher. You can enter the relevant information in the following fields:

- **Name** — The name is a unique name in the launcher.
- **Command** — This is the command line string that is used to invoke the external process. This command string can use the variables from the **Workspace**, **File System**, or **Variables** configured in Eclipse. These variables are replaced with their corresponding values when launched.

## Testing a subset of your apama application

In a large Apama application, you might want to test a subset instead of the entire application. The best way to do this is to define a launch configuration that injects only the monitors you want to test. Use this configuration only to test the subset. Create a different launch configuration to test the entire application.

The reason you want a separate configuration for testing the subset is that you must disable the monitors in your project that are not in the subset you want to test.

---

### To disable a file in a launch configuration

1. Define the launch configuration as you normally would. See ["Defining custom launch configurations" on page 99](#).
2. In the **Create, manage, and run configurations** dialog, on the **Components** tab, select the correlator in which you want to perform the testing and click **Edit**. Or, click **Add > Add Correlator** to add the correlator on which you want to do the testing.
3. In the **Correlator Configuration** dialog that appears, on the **Initialization** tab, deselect the file(s) you do not want to inject.
4. Click **OK**, **Apply**, and then **Run** or **Close**.

## Monitoring Apama applications

Software AG Designer provides various ways to monitor a running Apama application:

- Console view
- Engine Information view
- Engine Receive view
- Engine Status view
- Scenario Browser
- Dashboards


Note that in the Apama Developer perspective, it is possible to launch multiple projects at any one time. By default, the Console view and the Scenario Browser view will display information about the most recently launched correlator. To monitor a different correlator, select **Window > Show View > Other > Debug** from the menu and select the Debug view — this lists all running launches. Select the correlator that you wish to monitor from this list.

Opening the Apama Runtime perspective (see "[The Apama Runtime perspective](#)" on [page 17](#)) results in a convenient layout of all the available Apama Runtime views. This is the recommended way to make use of these views. You can switch back to the Apama Developer or Apama Workbench perspective at any time.

### Console view

The Console view displays information concerning a running Apama application. An application can have several consoles:

- **Correlator** — Displays output from the correlator.
- **Engine Inject** — Displays initialization information injected to the correlator.
- **Engine Send** — Displays information from Apama components such as dashboards that stream data to the correlator.
- **Correlator Initialization** — Displays information about the correlator initialization including the Java files, `.mon` files (monitors), and `.sdf` files (scenarios) that have been injected and the `.evt` files (events) that have been sent and whether the actions succeeded or failed.

To view one of these consoles, click the drop down arrow of the **Display Selected Console** button  and select the console you want.

## Using the Engine Information view

The Engine Information view inspects a running correlator and displays defined contents of the correlator. It displays the same information as the Apama command line tool `engine_inspect`.

The information is grouped as follows (click on the right-facing triangle to view the contents of each group):

**Aggregates** — The names of custom (user-defined) aggregate functions that have been injected. You use aggregate functions in stream queries. Apama built-in aggregate functions are not listed.

**Contexts** — The names of any user-defined contexts, how many monitor instances are running in each context, how many entries are on each context's input queue, and the names of any channels each context is subscribed to.

**Events** — The names of all event types in the correlator, and the number of templates of each type, as defined in listener specifications.


**Java Monitors** — The names of all Java applications in the event correlator and the number of event listeners each Java application has created.


**Monitors** — The names of all monitors in the event correlator and the number of monitor instances each monitor has spawned.

**Plugin Receivers** — The names of any plug-in receivers, the channels each plug-in receiver is subscribed to, and the number of items on each plug-in receiver's input queue. A plug-in receiver is a correlator plug-in that is subscribed to one or more channels.

**Receivers** — The names of any external receivers, the address of each external receiver, the channels each external receiver is subscribed to, and the number of entries on the output queue of each external receiver.

**Timers** — Displays the current EPL timers active within the system. The four types of timers that might be displayed here are `wait`, `within`, `at`, and `stream`. The `stream` timers are those set up to support the operation of a stream network.

This view allows direct deletion of the defined named entities using the **Delete** button  on the tool bar or from the context (right-click) menu.

The user can send an event from this view as well using the send operation by clicking the Send button  on the toolbar or via the context menu. Double clicking an event item also invokes this operation.

Displaying information in the Engine Information view can slow performance, so you may want to close the view if Software AG Designer is not very responsive.

## Using the Engine Receive view

The Engine Receive view shows all events generated from the connected correlator. Each batch of events is separated with an optional separator.

- **Select All** — Select all text in the console.
- **Clear** — Clear all text in the console.
- **Copy** — Copy all selected text to the clipboard
- **Show Separator** — Show a separator line in each event batch.
- **Show Date/Time of batch** — Show a date/time of each event batch

You can also change whether to display the separator line and change how it appears by clicking the drop down arrow at the top right of the view to display the menu and select **Options**. See ["Engine Receive Viewer preferences" on page 109](#).

Click the **Toggle Connection** button  to temporarily disconnect the Engine Receive view from the correlator.

When the **Toggle Connection** button is pressed, the console will not update any further events from correlator. The background of the console will be changed to indicate the temporary disconnect mode. The **Select All**, **Clear**, and **Copy** actions will still work in this mode.

To resume from the temporary disconnect mode, simply click on the **Toggle Connection** button again to resume the connection. The console will be cleared and new events will be shown in the console again.

## Engine Receive Viewer preferences

You can change the display options of the Engine Receive view by clicking the drop down arrow at the top right of the view. From the menu select **Options**, which displays the Engine Receive Viewer Preferences dialog. This dialog contains settings that specify how information appears in the Engine Receive view.

**Display a separator** — Toggles whether each batch of events is visually separated.

**Display date/time for batch** — Toggles whether the date and time for each batch of events appears.

**Custom text for separator** — Specifies the characters composing the separator.

**Max window size (KB)** — Select a setting between 100 and 10000.

## Using the Engine Status view

The Engine Status view displays the information about the correlator status. The information is the same as the output of Apama command line tool `engine_watch`.

**Uptime(ms)** — The time in milliseconds since this correlator was started. This figure is unaffected if the state of the correlator is restored from a checkpoint file.

**Number of contexts** — The number of contexts in the correlator. This includes the main context plus any created contexts.

**Number of monitors** — The number of monitor definitions injected into the correlator. This figure changes upwards and downwards as monitors are injected, deleted or just expire. A monitor expires when each of its instances dies, has no listeners left, or causes a runtime error.

**Number of sub-monitors** — The number of monitor instances across all contexts in the correlator. In monitors, spawn actions create monitor instances. This figure changes upwards and downwards as monitor instances are spawned, killed or just expire.

**Number of java applications** — The number of JMon applications loaded in the correlator. JMon applications do not expire, so this value only decreases when you explicitly unload a JMon application.

**Number of listeners** — The number of event listeners created by monitor instances across all contexts, plus the number of JMon applications.

**Number of sub-listeners** — The number of sub-listeners that have been created by listeners across all contexts.

**Number of event types** — The total number of event types defined within the correlator. This figure decreases when you delete event types from the correlator.

**Events on input queue** — The sum of the number of events on the input queue of each context. The main context has its own input queue and any user-defined contexts each have an input queue.

**Events received** — The total number of events ever received by the correlator. A checkpoint preserves this value. If you restore the state of the correlator from a checkpoint file, this number reflects the total number of the events seen by the correlator from which the checkpoint was originally made. Note that if an event is on an input queue, it has been received but not processed.

**Events processed** — The total number of events processed by the correlator in all contexts. This includes external events and events routed to contexts by monitors. An event is considered to have been processed when all listeners that were waiting for it have been triggered, or when it has been determined that there are no listeners for the event.

**Events on internal queue** — The sum of the number of routed events on the input queue of each context. The internal routing queue in each context is a high priority queue for events that you internally routed with the `route` instruction in your EPL files. The correlator always processes routed events before processing events on the input queue.

**Events routed internally** — The sum of the number of events routed in each context since the correlator started. A checkpoint preserves this value. If you restore the state of a correlator from a checkpoint file, this number reflects the total number of the events

routed to the internal queues for the correlator from which the checkpoint was originally made.

**Number of consumers** — The number of event consumers registered with the correlator. Event consumers receive events emitted by the correlator.

**Events on output queue** — The number of events waiting on the correlator's output queue to be dispatched to any registered event consumers.

**Output events created** — The total number of output events created by the correlator. A checkpoint preserves this value. If you restore the state of a correlator from a checkpoint file, this number reflects the total number of output events created by the correlator from which the checkpoint was originally made.

**Output events sent** — The total number of output events that the correlator has sent to event consumers. For example, suppose the correlator created 10 output events and sent each event to two consumers. The number of output events sent is 20. A checkpoint preserves this value. If you restore the state of a correlator from a checkpoint file, this number reflects the total number of output events sent by the correlator from which the checkpoint was originally made.

## Using the Scenario Browser view

Apama's Scenario Browser view displays the scenario and query definitions loaded into the correlator along with the metadata for those scenarios and queries. For queries, Scenario Browser displays parameterized and non-parameterized queries that are running along with the query parameter values. In the Scenario Browser you can add, edit, and delete scenario and parameterized query instances.

**Note:** A non-parameterized query contains a single default instance. You cannot edit or delete the default instance. Editing and deleting options are disabled for the default instance.

## Displaying the Scenario Browser

By default, in the Apama Workbench perspective, the Scenario Browser is opened when you launch an application. In the Apama Developer perspective, you need to display the Scenario Browser view manually by selecting **Window > Show View > Scenario Browser** from the menu.

Apama Developer perspective users may find it more convenient to switch to the Apama Runtime perspective (see ["The Apama Runtime perspective" on page 17](#)). You can switch back to the Apama Developer perspective at any time. The details of the selected correlator are shown in the right pane.


## Browsing scenarios and query definitions

To examine a scenario definition or a query definition, click the name of scenario definition or query definition in the left pane. The Scenario Browser displays the information in two tabs in the right pane:

- The **Details** tab displays the metadata of the selected scenario or query.
- The **Instance Summary** tab displays the output fields of the selected scenario. Each column represents a scenario output field.

## Creating a new instance of a scenario or a parameterized query

In the Scenario Browser view, you can create new instances of a scenario or a parameterized query that is running in the application.

1. In the Scenario Browser's left pane, select the scenario or query for which you want to create a new instance.
2. Click the **Add Scenario Instance** button . The Add Scenario Instance dialog appears, showing the details including the input and output variables that make up the scenario or query definition in the left-hand column. The right-hand column displays the values of those variables.
3. To create an instance of a scenario or parameterized query, you must specify values for the input fields in the right-hand column. Fields with default values are already filled in. You can add or edit values in cells that have a white background. To specify a value, double click in the empty cell and enter the information.
4. Click **OK**.

## Viewing the instances of a scenario or a query


You can display the details of a scenario instance or a query instance in the Scenario Browser.

1. In the Scenario Browser, select the instance ID of the scenario or the query. In the right pane, the names of the variables are shown in the left column and the current values of the variables in the right column. The values of the output variables are continuously updated.

You can expand and collapse the display in the right pane by clicking the plus and minus symbols.

## Editing an instance of a scenario or a parameterized query

### To edit the input values of a scenario instance or a parameterized query instance

1. In the left pane of the Scenario Browser, select the instance ID of the scenario or the query you want to edit.
2. Click the **Edit** button . The Edit scenario instance dialog appears. While you are editing the values in the dialog, Software AG Designer suspends the display of output information.
3. Edit the values for the input fields in the right-hand column by double clicking in the desired cell and entering the new value.
4. Click **OK**. Software AG Designer resumes displaying output values.




---

## Deleting an instance of a scenario or a parameterized query

---

### To delete a scenario instance or a parameterized query instance


1. In the left pane of the **Scenario Browser**, select the instance ID of the scenario or the query you want to delete.
2. Click the **Delete** button .

---

## Deleting all instances of a scenario or a parameterized query

---

### To delete all instances of a given scenario or a parameterized query

1. In the left pane of the **Scenario Browser**, select the name of scenario or query for which you want to delete all the instances.
2. Click the **Delete All** button .

## Dashboards

When you launch a project that contains a dashboard, by default Software AG Designer automatically starts the Dashboard Viewer and displays the project's default dashboard. If desired you can change this behavior as described in "[Defining custom launch configurations](#)" on page 99. You can specify that you do not want Dashboard Viewer to start automatically or that Viewer should display a dashboard other than the project's default dashboard. When you define a launch configuration you can also specify any command line options to need to be passed to the Dashboard Viewer.



# 5

## Debugging EPL Applications

---

■ Adding breakpoints .....	116
■ Launching a debug session .....	116
■ Debugging a remote application .....	117
■ Debug view for an EPL application .....	118
■ Breakpoints view for an EPL application .....	118
■ Variables view for an EPL application .....	119
■ Command-line debugger .....	119

You can debug Apama applications written in EPL in Software AG Designer. As is the case when debugging other applications in Eclipse, when you debug EPL applications, you can set breakpoints to suspend the application at specific points, examine the contents of variables, and step through the application.

The basic tasks involved in debugging an EPL application in Software AG Designer are:

1. Set breakpoints in the Developer perspective.
2. Create a debug launch configuration for the project.
3. Review the executing application in the Debug perspective, examining information such as the call stack, context status, and variable values.

## Adding breakpoints

---

You can add breakpoints to any of the EPL files included in the project. To add a breakpoint to an EPL file:

1. In the Developer perspective open the EPL file in which you want to add a breakpoint.
2. Double click in the left margin of the desired line in the EPL file or right-click in the left margin and select **Toggle Breakpoint** from the pop-up menu. The breakpoint is indicated by a solid blue circle in the left margin.
3. Repeat Step 1 and Step 2 for each breakpoint you want to set.

Note, when debugging an application on a remote machine, breakpoints are supported only inside EPL actions of monitors and events. Breakpoints inside listeners of actions or breakpoints outside of actions are not supported. See ["Debugging a remote application" on page 117](#).

## Launching a debug session

---


If a Debug Configuration has been defined for the Apama project, start a debugging session as follows. (If you need to create a Debug Configuration, see ["Creating a debug configuration" on page 117](#))

1. From the Software AG Designer menu, select **Run > Debug Configurations**.
2. From the Debug Configuration dialog, select the desired Debug Configuration and click Debug. The Confirm Perspective Switch dialog appears.
3. Click **Yes** to display the Debug Perspective (and add a check to the **Remember my decision** check box if desired). The Debug perspective appears.

When debugging an EPL application, the Debug perspective is similar in operation to the standard Eclipse Debug perspective and includes the Debug, Breakpoints, and Variables views.

## Creating a debug configuration

### To create a new Debug Configuration

1. From the Software AG Designer menu, select **Run > Debug Configurations**. The Profile configurations wizard starts.
2. In the left pane, select **Apama Application** and click the **New launch configuration** button ()
3. In the right pane, on the **Apama Project** tab, specify the **Name** and the **Project**.
4. If you are going to debug an application running on a remote correlator:
  - a. Select the **Components** tab.
  - b. Click **Add** and select **Add Correlator**.
  - c. In the Correlator Configuration wizard, on the **Correlator Arguments** tab, specify the **Host**, **Port**, and other **Options**.

**Note:** When creating a debug launch configuration, you do not need to specify the `-g` option in the **Correlator Arguments** tab. When you select **Run > Debug**, Software AG Designer automatically starts the correlator with the `-g` option, which disables optimizations that hinder debugging. However, if you use Software AG Designer to debug in a remote correlator, you must ensure that the remote correlator was started with the `-g` option. You cannot debug in a correlator that was started without specification of the `-g` option.

See options information in "Starting the event correlator" in *Deploying and Managing Apama Applications*.

- d. If necessary, specify any initialization options on the **Initialization** tab.
  - e. Click **OK**.
5. Click **Debug**. The Confirm Perspective Switch dialog opens.
  6. Click **Yes**. Software AG Designer switches to the Debug perspective.


## Debugging a remote application

In Software AG Designer, you can debug an Apama application running in a correlator on a remote machine. The correlator on the remote machine must have been started with the `-g` option, which disables optimizations that interfere with debugging.

**Note:** When debugging a remote application, breakpoints are supported only inside actions of monitors and events. Breakpoints inside listener actions or outside of actions are not supported.

---

### To debug an application on a remote machine

1. Start the application on the remote machine.
2. In Software AG Designer on the local machine where you will do the debugging, select **Run > Debug Configurations**. This displays the Debug Configurations wizard.
3. In the Debug Configurations wizard, in the left pane select **Remote Apama Application**, and click the **New launch configuration** button ().
4. In the right pane, on the **Apama Project** tab, specify the following:
  - a. The **Name** of the debug configuration and the **Project** on the local machine where the debugging session is being run. The application in the project should match the application that is being debugged.
  - b. The **Host** and the **Port** of the correlator running on the remote machine.
  - c. If desired, click **Test Connection** to verify that the connection to the correlator on the remote machine can be made.
5. If desired, on the **Source** tab, add any needed resources. Generally, the project on the local machine will contain all the necessary resources.
6. Click **Debug**. The Confirm Perspective Switch dialog opens.
7. Click **Yes**. Software AG Designer switches to the Debug perspective.

---

## Debug view for an EPL application

The Debug view shows the call stack and status of current contexts in the Apama application. When execution reaches a breakpoint, the current context in the Debug view is highlighted.

Also, when a breakpoint is reached, the EPL editor indicates the breakpoint in the code with an arrow in the left margin.

When a stack frame is highlighted in the Debug view, the Variable view displays the names and values of the associated variables, see "[Variables view for an EPL application](#)" on page 119.

See the Eclipse online help for detailed information on this view.

---

## Breakpoints view for an EPL application

The Breakpoints view lists the breakpoints you have set in the project.

Double-clicking a breakpoint displays the line in the EPL file that contains the breakpoint in the EPL editor. In the Breakpoints view, you can enable and disable the listed breakpoints.


See the Eclipse online help for detailed information on this view.


## Variables view for an EPL application

---

The Variables view displays information about the variables associated with the stack frame that is currently selected in the Debug view.

The variables can be expanded to show their fields. The detail pane in the area at the bottom of the view displays text. For a variable of a primitive type, this is the value of the object. For a complex variable, such as a sequence, dictionary, or event, it is the text representation of the object as returned by the object's `toString()` method.

For complex variables, you can click the **Show Logical Structure** button () to expand the variables to show their sub-values.

If you want to change the layout of the display, click the drop-down menu button () at the top right of the Variables view. From the resulting menu, select **Layout** and then the desired layout option.

See the Eclipse online help for detailed information on this view.

## Command-line debugger

---

You can also use Apama's `engine_debug` tool to control execution of your EPL code in the correlator and inspect correlator state. This tool is a correlator client that runs a single command from the command line. It is not an interactive command-line debugger. The interactive debugger in Software AG Designer is more useful during the development of an application. In general, the command-line `engine_debug` tool is expected to be most useful during or after deployment of your application.

For more information on `engine_debug`, see "Using the command line debugger" in *Deploying and Managing Apama Applications*.





# 6 Debugging JMon Applications

---

■ Preparing the correlator for remote debugging .....	122
■ Creating a debug run configuration .....	123
■ Debug perspective .....	124
■ Example debug session .....	125
■ Additional resources for Java debugging .....	127

This section describes practical information for developing and debugging JMon applications with Software AG Designer. General knowledge of Java and Apama application development is assumed.

Software AG Designer is built on the Eclipse IDE framework and as such running and debugging JMon applications for the Apama correlator engine is no more different than with any standard Java application. There are however a few things to consider:

- **Single thread** — For JMon applications, the correlator uses a single thread of execution. When programming in Java consider the importance of maintaining determinism by not routing, emitting, or enqueueing an event in a thread other than the current thread. Java provides no guarantee on execution order for separate threads, so try to keep your JMon applications single threaded at all times. Although multiple threads are currently allowed for JMon applications, this might change in the future, with the correlator JVM issuing a warning or possibly an error.
- **Event definitions** — To instantiate event definitions in your JMon application, the `Event` subclass in question needs to be defined and included in your `JAR` file. This is because the correlator uses a separate classloader for each application (that is, each JMon `JAR` file injected) and hence cannot share the event definitions across separate JMon applications. Also, a JMon application cannot make use of any event definition already present in the correlator. Any event definition (either a subclass of the JMon `Event` class or a definition in an EPL file) must be replicated for each JMon application and for your injected EPL files.

## Preparing the correlator for remote debugging

---

Launching the correlator with the `-j` option enables the Java Virtual Machine (JVM) as a subprocess in correlator execution. When launching an Apama project in Software AG Designer, specification of the `-j` option launches the correlator and launches its JVM as an external process. Consequently, you must enable a socket connection to the JVM so that you can connect the Software AG Designer debugger to it. This also applies to launching a correlator through any other means.

Enable debugging in the correlator JVM by specifying the Java options for the correlator component in the launch configuration. The Java Debug Wire Protocol (JDWP) is the protocol used for communication between a debugger and the Java virtual machine (VM) that it debugs. Oracle's VM implementations require command line options to load the JDWP agent for debugging. JDWP is optional; it might not be available in some implementations of the Java™ 2 SDK. The existence of JDWP can allow the same debugger to work.

From Java 5.0 (or 1.5.0) onwards, specify the `-agentlib:jdwp` option to load and specify options to the JDWP agent. For Java releases prior to 5.0, the `-Xdebug` and `-Xrunjdwp` options are used. (The 5.0 implementation also supports the `-Xdebug` and `-Xrunjdwp` options but the newer `-agentlib:jdwp` option is preferable as the JDWP agent in 5.0 uses the JVMTI interface to the VM rather than the older JVMDI interface.) Specify this option in the correlator component in your launch configuration or when launching the correlator independently:

```
-J -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n
```

This tells the JVM to use a socket transport on port 8787. You can change this at will and use your own options. For more Java debugging/agent options, see <http://docs.oracle.com/javase/1.5.0/docs/guide/jpda/conninv.html#Invocation>.

To launch the correlator with Java and remote debugging enabled, you can use a command line or Software AG Designer.

On the command line, invoke the correlator with the following arguments:

```
correlator -l ApamaServerLicense.xml -j -J
-agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n
```

In Software AG Designer, invoke the correlator as follows:

1. From the Software AG Designer menu, select **Run > Run Configurations**.
2. Click the **Components** tab.
3. Select the correlator you want to start and click **Edit**.
4. Enter the following in the **Extra command line arguments** field:

```
-J -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n
```


5. Select **OK** and then **Run** to start the correlator.

The correlator is now enabled for Java debugging. The next section discusses how to connect to the correlator with the Software AG Designer debugger.

## Creating a debug run configuration

After you have a running correlator that is enabled for remote debugging, you must create a debug run configuration that you use to connect to the correlator:

### To create a debug run configuration

1. From the Software AG Designer menu, select **Run > Debug Configurations**.
2. In the Debug dialog, on the left, select **Remote Java Application**.
3. From the Debug dialog toolbar, select **New launch configuration** .
4. Select the **Connect** tab, if it is not already selected.
5. Enter a name for your project and set the host and port for the remote JVM connection you previously specified as Java options when you started the correlator.
6. Click the **Source** tab and add the path to the Java source for your JMon application.
7. Click **Apply** and then **Debug**.

This launch configuration immediately connects to the remote JVM process in the correlator and Software AG Designer switches to the Apama Runtime perspective, which you use for debugging. You are now ready to remotely debug your JMon

application. Debugging a JMon application in Software AG Designer is the same as debugging any other Java application in Eclipse.

## Debug perspective

---

The Debug perspective appears automatically when you start a debug session. The default Debug perspective has five panels that you can use for debugging:

- The top-left panel is the Debug view which shows the application's stack frame. See ["Using the Debug view with a JMon application" on page 124](#).
- The top-right panel provides a tabs for viewing the Breakpoints view and the Variables view. See ["Working with breakpoints in a JMon application" on page 124](#) and ["Viewing stack frame variables for a JMon application" on page 125](#).
- The middle-left panel is the code editor view.
- The middle-right panel is the code Outline view.
- The bottom panel includes the standard tabs for the console output, tasks, errors, problems, search results, display, and so on.

## Using the Debug view with a JMon application

The Debug view shows the call stack and status of current threads, including any threads that have already run to completion. Whenever execution reaches a breakpoint, the current thread in the Debug view is highlighted.

This view provides code execution controls. These allow you to suspend and resume execution, step through breakpoints in various ways, jump between stack frames, connect/disconnect to remote JVMs process, and so on. See the Eclipse online help for detailed information on this view.

## Working with breakpoints in a JMon application

You can set breakpoints in your code in a number of ways. The most straightforward way is to select a line in your Java code and double-click on the gray area to the left of the line. This sets a breakpoint before the execution of that line. The debugger will suspend execution when it reaches that line, allowing you to inspect variable values and results of expressions, either as defined in the code or created by you in real-time. These expressions can also be used in conditions for breakpoints.

You can set breakpoints on particular Java exceptions, on particular line numbers, on particular method calls, on conditions such as after a certain number of hits on a particular statement, and so on.

The Breakpoints view allows you to view and control all your currently defined breakpoints. After execution reaches a breakpoint, you can control process execution by

stepping forwards or backwards through your code execution, or by simply resuming execution until the next breakpoint or until your program terminates.

See the Eclipse online help for detailed information on this view.

## Viewing stack frame variables for a JMon application

When selecting a particular stack frame in the Debug view, the Variables view lets you inspect all variables as defined in the current stack context. Of course, you can also view variables alongside other variable values (globals, constants, and so on) if they are specified in the options. See the Eclipse online help for detailed information on this view.

## Example debug session

---

This section provides a very simple example that highlights some of the basic debugging concepts. It shows the basic steps for creating a JMon JAR file and injecting it into a debug enabled correlator. Familiarity with how to develop JMon applications is assumed and some steps are omitted based on this assumption.

---

### To develop a JMon application

1. Implement the Java code.
2. Annotate the Java code with required JMon information.
3. Generate the classes from the code.
4. Package the classes into a JAR file.
5. Create the JMon JAR manifest manually or by using the `JarProcessor` utility.
6. Inject the JAR file into a running correlator that is enabled with the JVM and with the proper Java options for remote debugging.

## Debug example of preparing code and JAR file

Consider the following sample Java code, which is the basis for a JMon application. The files containing the complete code are `Simple.java` and `Tick.java`, which are located in the `samples\java_monitor\simple\src` directory of your Apama installation.

```
@Application(name = "Simple",
             author = "John Doe",
             version = "1.0",
             company = "Apama",
             description = "sample",
             classpath = ""
            )
@MonitorType(description = "A simple Java monitor")

public class Simple implements Monitor, MatchListener {
    public Simple() {}
    public void onLoad() {
```

```

    EventExpression eventExpr =
        new EventExpression("all Tick(*, >10.0):t");
    eventExpr.addMatchListener(this);
}

public void match(MatchEvent event) {
    Tick tick = (Tick)event.getMatchingEvents().get("t");
    tick.emit();
}
}

```

Here is the Tick class (Tick.java):

```

@EventType(description = "Event that signals a stock trade")
public class Tick extends Event {
    public String name;
    public double price;
    public Tick() {
        this("", 0);
    }

    public Tick(String name, double price){
        this.name = name;
        this.price = price;
    }
}

```

Assume that Simple.java is part of your Apama project. Open or create the project in Software AG Designer and add the code if necessary. Note that this code is already annotated with the information required for creating the JMon JAR manifest.

The next step is to compile and pack the class bytecode into a JAR file, and create the manifest that flags and describes this JAR file as a JMon application. You can do this by setting up a new builder in the project properties in Software AG Designer or through the command line. For the sake of simplicity, the standard command line calls are described below:

1. Compile the Java code. For example:

```

javac -classpath "%APAMA_HOME%/lib/ap-correlator-extension-api.jar;
%APAMA_HOME%/lib/ap-util.jar" *.java

```

2. Package the classes into your JAR file:

```

jar cf simple-jmon.jar *.class

```

3. Use the JarProcessor utility to create the manifest for the JAR file. For example, if the JAR is called simple-jmon.jar, enter:

```

java -classpath "%APAMA_HOME%/lib/ap-correlator-extension-api.jar"
com.apama.jmon.annotation.JarProcessor simple-jmon.jar

```

## Debug example of starting correlator and injecting application

Start a local correlator with remote debugging enabled. For example, if you specify port 8787, the command line looks like this:

```

correlator -l ApamaServerLicense.xml -j -J
-agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n

```

Inject the JAR into the running correlator. For example:

```
engine_inject -j simple-jmon.jar
```

You are ready to start an Apama debug session.

## Example of debugging in Software AG Designer

You want to connect to the remote correlator and debug the JMon application in Software AG Designer. To do this, you must open the JMon project (or the Apama project with Java support enabled) in Software AG Designer and create a debug launch configuration. This launch configuration connects to the remote Java application, which is the JVM in the running correlator.


For test purposes, suppose that you create a simple breakpoint in the `match()` method in `Simple.java`. This suspends execution when the correlator reaches a matching event listener.

Send the following events to the correlator to trigger the breakpoint:

```
Tick("ibm", 1.0)
Tick("ibm", 5.0)
Tick("ibm", 15.0)
Tick("msft", 15.0)
```

As you can see, the debugger stops execution at the specified code breakpoint after the listener finds the first `Tick` event with a price greater than 10.0, that is `Tick("ibm", 15.0)`.

Suppose that you want to examine the heap context, that is, the values of the variables. You can observe this in the top left **Debug** panel of the Apama Runtime perspective. Select the **Simple.match** method stack frame from **Thread[ApamaProcessing]**.

Note that the Variables view now shows the proper stack frame context with all relevant heap space variable values. The `tick` variable, defined in the code, is not yet visible. This is because execution was suspended before the current line was executed. To execute the current line, which extracts the matching `Tick` event and assigns it to the `tick` variable, click **Step Over**  in the **Debug** panel toolbar. As you can see, the `tick` variable now appears in the Variables view. You can select it to inspect its value, which is, of course, `Tick("ibm", 15.0)`.

## Additional resources for Java debugging

This section introduces the basic concepts for debugging JMon applications in Software AG Designer. There are a number of Software AG Designer debugging features not mentioned, but the mechanisms are essentially the same as for other Java applications. For additional information about available debugging options, see the following Eclipse information:

- Running and debugging Java in Eclipse:

<http://help.eclipse.org/kepler/index.jsp?topic=/org.eclipse.jdt.doc.user/>

[tasks/task-running\\_and\\_debugging.htm](#)

- Eclipse And Java: Using the Debugger (Video Tutorial):  
<http://www.eclipse.org/resources/resource.php?id=405>
- JDB (Java debugger):  
<http://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/jdb.html>
- Good video tutorials and practice exercises for debugging Java in Eclipse:  
<http://eclipsetutorial.sourceforge.net/debugger.html>
- IBM debugging with Eclipse:  
<http://www.ibm.com/developerworks/library/os-ecbug/>



# 7

## Profiling EPL Applications

---

■ Launching profiling sessions .....	130
■ The Apama Profiler perspective .....	133
■ Using filters .....	135
■ Taking snapshots .....	136
■ Using snapshots .....	137
■ Choosing profiling information columns .....	138
■ Updating profile data .....	138
■ Displaying Apama perspective preferences .....	138

You can profile Apama applications written in EPL with Software AG Designer. Data collected in the profiler allows you to identify possible bottlenecks in an EPL application. The profiler consists of the Apama Profiler perspective which includes a variety of views. With Apama's EPL Profiler, you can profile applications running on both local and remote correlators.

**Note:** You can also profile Apama applications using the `engine_management` tool. See "Using the CPU profiler" in *Deploying and Managing Apama Applications* for further information.

---

## Launching profiling sessions

You can profile an Apama EPL application using a default launch configuration. You can also create custom profile launch configurations for launching profiling session. For example, you may want the profiler to launch the application using a correlator different from the project's default correlator or to profile an application running on a remote correlator.

- When a default profile configuration is launched, it starts the profiler and the default correlator and the application if they aren't already running. If they are already running the profiler just starts profiling the running application. See "[Launching a default profiling session](#)" on page 130 for more information on launching a default profiling session.
- If you want to profile an application that runs on a non-default correlator, you need to create a launch configuration that points to the run configuration associated with the desired correlator. See "[Launching a custom profiling session](#)" on page 131 for more information for creating and launching a custom profiling configuration.
- If you want to profile an application that runs on a remote correlator, you need to create a launch configuration that points to the machine where the correlator is running. In order to profile an application running on a remote correlator, the remote application needs to be running before you launch the profiling session. See "[Launching a remote profiling session](#)" on page 132 for more information on creating and launching a configuration for a remote profiling session.

## Launching a default profiling session

You can profile an Apama EPL application using a default launch configuration or you can create a custom profiler launch configuration for an application running on a remote correlator.

---

### To launch a profiling session in Software AG Designer using the default launch configuration

1. From the Software AG Designer menu, select **Run > Profile**.
2. By default, Software AG Designer displays the Confirm Perspective Switch dialog asking if you want to use the Apama Profiler perspective. Click **Yes**. Software AG

Designer switches to the Apama Profiler perspective and begins collecting data from the EPL application.

Note, you can change the behavior of the Confirm Perspective Switch dialog by changing a setting in the Profiling Monitor view's **Preferences** wizard; see ["Displaying Apama perspective preferences" on page 138](#).

## Launching a custom profiling session

You can create custom launch configurations for profiling Apama EPL applications. See ["Creating a custom profile launch configuration" on page 131](#) for more information on creating a custom profiler launch configuration.

---

### To launch a profiling session in Software AG Designer using a custom launch configuration

1. In the Project Explorer right-click the project name and select **Profile As > Apama Correlator Profiler**. The Select Profile Application dialog appears.
2. In the Select Profile Application dialog, select the launch configuration you want to use and click **OK**.
3. By default, Software AG Designer displays the Confirm Perspective Switch dialog asking if you want to use the Apama Profiler perspective. Click **Yes**. Software AG Designer switches to the Apama Profiler perspective and begins collecting data from the EPL application.

Note, you can change the behavior of the Confirm Perspective Switch dialog by changing a setting in the Profiling Monitor view's **Preferences** wizard; see ["Displaying Apama perspective preferences" on page 138](#).

## Creating a custom profile launch configuration

You can create custom profile launch configurations for launching profiling session. For example, you may want the profiler to launch the application using a correlator different from the project's default correlator.

---

### To create a custom profile launch configuration

1. In the Project Explorer right-click the project and select **Profile As > Profile Configurations**. The Profile Configurations wizard starts.
2. In the Profile Configurations wizard, click the **New launch configuration** button.
3. In the Profile Configurations wizard, replace the default name in the **Name** field if desired.
4. In the Profile Configurations wizard, on the **Connection Details** tab, in the **Profile a** field, use the default **Apama Launch configuration** selection.
5. In the **Launch Configuration** field, click the **Browse** button. The Choose Launch Configuration dialog appears.

6. In the Choose Launch Configuration dialog, select the run configuration you want to profile and click **OK**.
7. In the Profile Configurations wizard, click **Apply** to save the profile configuration or **Profile** to save and launch the profiling session.

## Launching a remote profiling session

In Software AG Designer, you can profile an Apama application running on a remote correlator by creating a custom profiler launch configuration that points to the remote machine where the correlator is running.

Before you launch a profiling session for an application running on a remote correlator, the remote application needs to be already running, and you must also create a remote profiler launch configuration.

---

### To launch a profiling session for an application on a remote correlator

1. In the Project Explorer right-click the project name and select **Profile As > Apama Correlator Profiler**. The Select Profile Application dialog appears.
2. In the Select Profile Application dialog, select the launch configuration you want to use to connect to the remote correlator and click **OK**.
3. By default, Software AG Designer displays the Confirm Perspective Switch dialog asking if you want to use the Apama Profiler perspective. Click **Yes**. Software AG Designer switches to the Apama Profiler perspective and begins collecting data from the EPL application.

Note, you can change the behavior of the Confirm Perspective Switch dialog by changing a setting in the Profiling Monitor view's **Preferences** wizard; see "[Displaying Apama perspective preferences](#)" on page 138.

## Creating a remote profiler launch configuration

---

### To create a remote profiler launch configuration

1. From the Project Explorer, right-click the project and select **Profile As > Profile Configurations**. The Profile Configurations wizard appears.
2. In the Profile Configurations wizard, enter a meaningful name for the configuration in the **Name** field if you do not want to use the default name.
3. On the **Connection Details** tab, in the **Profile a** field, select **Remote Apama correlator**.
4. In the **Remote Correlator Details** section, enter the name of the host and the port in the **Host** and **Port** fields.
5. If desired, and if the remote correlator is running, click **Test Connection** to confirm the specified location information is correct.

6. Click **Apply** to save the profile configuration or **Profile** to save and launch the profiling session.

## The Apama Profiler perspective








The Apama Profiler perspective consists of the following views:

- Profiling Monitor view
- Execution Statistics view

### Profiling Monitor view

The Profiling Monitor view shows a tree view of the available profiler sessions. These sessions could be associated with different applications or associated with applications running on different correlators.

The tool bar for the Profiling Monitor view contains the following buttons:


-  — Resume the profiling session.
-  — Pause the profiling session.
-  — Stop the profiling session.
-  — Manually refresh the Execution Statistics view with collected data. By default the data is automatically refreshed at 10 second intervals; you can change the refresh behavior with the **Preferences** button described below.
-  — Displays a page of the Apama preferences on which you can change the profiler settings. For more information, see the description of the Apama preference page ["Profiling and Logging" on page 158](#).
-  — Collapse the entries in the tree view displayed in the Profiling Monitor view.
-  — Display the Profiling Monitor view's drop down menu.







### Execution Statistics view

The Execution Statistics view displays the timing details of the profiled EPL application. This view includes the following tabs:


- **Execution Statistics**
- **Comparison of Execution Statistics**


You can adjust the way the information appears using the buttons on the view's tool bar. The following tools are available:

-  — Apply and manage filters for displaying subsets of the profiling data. This tool is not available for the **Comparison of Execution Statistics** tab. For more information on filtering the data, see ["Using filters" on page 135](#)

-  — Expand the entire of tree of entries.
-  — Collapse the entire of tree of entries.
-  — Take a snapshot of the profiling data.
-  — Manage snapshots.
-  — Specify how to organize the profiling information by Contexts, Monitors/Events, or Action.
-  — Choose which columns to display. You can also change the order in which the columns are displayed.


## The Execution Statistics tab

The **Execution Statistics** tab carries a sub-title of "Session Summary". By default the left-hand column displays a tree view of the application's contexts, monitors, and actions organized by contexts. You can change the display to show profiler information organized by monitors/events and by actions using the **Organize View By** button [  ].

You can filter the profiler information in order to focus on specific application behavior using the **Filter** button [  ]. For more information on using filters, see ["Using filters" on page 135](#).

By default the display includes the following three columns of information:

- **CPU time** — The time in seconds this action has been executing.
- **Cumulative time** — The time in seconds spent in this action and all its child actions.
- **Idle** — The time in seconds this context has been idle, waiting for events.

You can also display the following information, by using the **Choose Columns** button [  ].

- **Empty** — The time in seconds this context has been empty, that is, without monitors.
- **Non-Idle** — The time in seconds this context has been non-idle, in other words, when it has had events to process.
- **Runnable** — The time in seconds this context has had work to do, but during which the work has not been executed by the scheduler.
- **Plugin** — The time in seconds spent in a plug-in call.
- **Blocked** — The time in seconds this context has been blocked, for example, waiting for a queue to become non-full.



## Comparison of Execution Statistics tab

This tab compares the most recent profiling data with that of a previous set. For each type of information there are three associated columns, for example **CPU**, **Old CPU**, and **Diff CPU**. The leftmost column is the current information, the column labelled "Old" is the


information from the previous set, and the column labelled "Diff" is the change between the new and the previous information. The previous set is considered the baseline.

By default, the "previous set" consists of the data from the last time it was refreshed. You can change this behavior to compare the most recent data to data captured in a snapshot at a particular moment in time. For information on capturing data in a snapshot, see ["Taking snapshots" on page 136](#)

As with the **Execution Statistics** tab, you can change the type of information and how it appears on the **Comparison Execution Statistics** tab as follows.

- Use the **Organize View By** button [  ] to organize the profiler information by Contexts, Monitors/Events, or Actions.
- Use the **Choose Columns** button [  ] to add, remove, or change the order of the information columns.

## Viewing EPL code

In the Execution Statistics view you can display the EPL code for a specific action or listener. Only the action-level entries that are shown with the code icon [  ] are able to display the associated code. There are other entries, such as garbage collector timings (that is, GC-mark, GC etc.), events and others for which there is no code association.

To display the code for an action or listener:

In the Execution Statistics view, double-click the action or listener for which you want to view the code. The EPL source file containing the code for the action or listener opens (if it is not already open) in the EPL editor.



## Using filters

When viewing the **Execution Statistics** tab of the Execution Statistics view, you can focus on specific data by applying a filter to the profiling information.

## Creating a Filter

You can create filters to concentrate on particular profiling data.

### To create a filter

1. On the **Execution Statistics** tab, click the **Filter** button [  ] on the view's tool bar and select **Manage Filters**. The Manage Filters dialog appears.
2. In the Manage Filters dialog, click **Add**. The Create Filter dialog appears.
3. In the Create Filter dialog, specify the **Filter Name** and an optional **Filter Description**.
4. In the Create Filter dialog, click the **Add New Row** button [  ]. This adds a row in the table field as indicated by a new set of parenthesis marks ( ).

5. Specify the filter criteria as follows:
  - a. Click the **Type** column and from the drop-down list select the type of data you want to filter for.
  - b. Click on the **Operation** column and from the drop-down list specify the operation that is appropriate to the Type you are filtering for.
  - c. Double-click in the **Value** column and enter an appropriate value; then click in a blank area of the table to accept the value you specified.
  - d. You can add other criteria to the filter by using AND or OR. In the **AND/OR** column, select the appropriate operator from the drop-down list, click the **Add New Row** and specify the additional filter criteria as described above.
6. Click **OK**.
7. In the Manage Filters dialog, select the filter you want and click **Apply**.


The **Execution Statistics** tab is updated according to your filter criteria.


## Managing Filters

Use the **Filter** tool to manage the use of filters. You can add, edit, or remove filters; change which filter is used; and revert to the default display of unfiltered data.

---

### To manage filters

1. On the **Execution Statistics** tab, click the **Filter** tool [  ] and select **Manage Filters** from the pop-up menu. The Manage Filters dialog appears.
  - If you want to create a new filter click **Add**. The Create Filter dialog appears. See ["Creating a Filter" on page 135](#) for more information on creating filters.
  - If you want to edit an existing filter click, select the name of the filter and click **Edit**. The Edit Filter dialog displays the current filter criteria. Edit the filter information as necessary and click **OK**.
  - If you want to remove a filter, select the name of the filter and click **Delete**.
2. Select the filter you want and click **Apply**. If you remove all the filters and click **Apply**, the profiler displays shows the default unfiltered set of information.

If you are looking at a filtered data set and want to return to the default display of unfiltered data, click the **Filter** tool [  ] and select **Reset Filter** from the pop-up menu.


## Taking snapshots

---

You can capture profiling data at any moment in time by taking snapshots. You can then compare current profiling data to the data represented in a snapshot.



### To take a snapshot

1. In the Execution Statistics view, click the **Take Snapshot** button . The Create Snapshot dialog appears.
2. Enter a meaningful name for the snapshot in the **Snapshot Name** field.
3. Specify the source of the data you want to use to create the snapshot as follows:

- To populate the snapshot with data that the correlator captures at this point in time, select **Current correlator**.
- To populate the snapshot with data that has previously been stored on disk in a .csv file, select **Select from file**. This data would typically be collected using the `engine_management` tool as, for example:

```
engine_management -r profiling get > myfile.csv
```

Note, however to be used this way, the .csv file must have been generated using Apama release 5.0.1 or later.

For more information on the `engine_management` utility, see "Shutting down and managing components" in *Deploying and Managing Apama Applications*.


4. Click **OK**.

In the **Comparison of Execution Statistics** tab, the baseline columns (labelled "Old") display the data captured in the snapshot and the columns labelled "Diff" are similarly updated to compare the current information with that captured in the snapshot. For more information on managing snapshots, see ["Using snapshots" on page 137](#).

## Using snapshots

After you create a snapshot the display on the **Comparison of Execution Statistics** tab is updated to compare the current data with that captured in the snapshot. You can change this to compare the current data to another snapshot or the default setting that compares the current data to the data seen the previous time the data was refreshed.

### To change the compared sets of data


In the **Comparison of Execution Statistics** tab, click the **Show Previous Snapshots** button .

- If you want to use the data from another snapshot, select the name you assigned to it.
- If you want to return to the profiler's default comparison, select **Compare last data**.
- If you want to manage your snapshots
  1. Select **Manage snapshots**.
  2. Select the name of the snapshot you want to apply or remove.

3. Click **OK** to apply the snapshot or **Remove** to remove the snapshot.
- If you want remove all snapshots, select **Clear Snapshots**.

## Choosing profiling information columns

You can specify which information to display in the Execution Statistics view. You can also change the order in which the columns are displayed. By default, the following columns are displayed:

- CPU — The time in seconds this action has been executing.
  - Cumulative — The time in seconds spent in this action and all its child actions.
  - Idle — The time in seconds this context has been idle, waiting for events.
1. In the Execution Statistics view, click the **View menu** button [  ] and select **Choose columns**. The Choose Columns dialog appears.
  2. Add a check to the check boxes that correspond to the columns you want to display.
  3. Click **OK**.

The Execution Statistics view is updated to show all the selected the columns.

## Updating profile data

By default, the Apama profiler polls for data every 10 seconds and then updates the display in the Execution Statistics view.

You can change the polling frequency in the Apama preferences, using the **Refreshing views** option. You can also change the way the profiler updates the data from a polling mode (the so-called automatic refresh) to an on-demand mode (the so-called manual refresh). For more information, see the description of the Apama preferences page ["Profiling and Logging" on page 158](#).

**Note:** You can directly access the appropriate Apama preferences page by clicking the **Preferences** button () in the Profiling Monitor view.

## Displaying Apama perspective preferences

By default, when you launch a profiling session, the Confirm Perspective Switch dialog is shown asking if you want to switch to the Apama Profiler perspective. You can change this behavior in the Apama preferences. For more information, see the description of the Apama preferences page ["Profiling and Logging" on page 158](#).

**Note:** You can directly access the appropriate Apama preferences page by clicking the **Preferences** button () in the Profiling Monitor view.



# 8

## Using the Data Player

---

■ Introduction to the Data Player .....	142
■ Using the Data Player .....	142
■ Data Player Control view .....	147
■ Creating Data Player query templates .....	148
■ Command-line Data Player interface .....	149

With the Apama Data Player you can play back previously saved event data as you develop your application. During playback, you can analyze the behavior of your application. Or, if you modify the saved event data, you can analyze how your application performs with the altered data. Apama plays back event data that has been stored in standard data formats.

## Introduction to the Data Player

---

The Data Player relies on Apama Database Connector (ADBC) adapters that are specific to standard ODBC and JDBC database formats as well as the proprietary Apama Sim format. These adapters run in the Apama Integration Application Framework (IAF), which connects the data sources to the correlator.

For more information, see "Using the Apama Database Connector" and "The Integration Adapter Framework" in *Connecting Apama Applications to External Components*.

The Apama Data Player consists of both the Query Editor and the Data Player Control.

## Using the Data Player

---

In addition to the normal operations for running an application in Software AG Designer, in order to play back event data in the Data Player, you need to perform a few other steps. Broadly, these steps are:

1. Configure the Apama project to use the appropriate ADBC adapter for the data source and database and specify the event types that will be played back along with the appropriate IAF mapping.
2. Launch the project so it can establish a connection to the data source.
3. Specify a Data Player playback query to determine what data from the database you want to play back.
4. Use the Data Player control to specify the following: how fast you want to play back the data; over what time range; and what throttling period to use.
5. Run the playback session.

## Adding the ADBC adapter

If you want to use the Apama Data Player in your project, you need to add and configure the Apama Database Connector adapter that is appropriate to the data source used by the project: ODBC, JDBC, or Sim.

---

### To add and configure an ADBC adapter

1. There are two ways of adding an ADBC adapter to a project.

**If you are creating a new Apama project:**

- a. Select **File > Project > New > Apama**.
- b. Give the new project a name, and click **Next**.

**If you are adding an ADBC adapter to an existing project:**

- a. In the **Project Explorer**, right-click the project and select **Apama > Add Adapter**. The Add Adapter Instance dialog opens.
  - b. If desired, in the Add Adapter Instance dialog, create a new name for the adapter instance or accept the default instance name. Software AG Designer prevents you from using a name that is already in use.
2. In the New Apama Project dialog or the Add Adapter Instance dialog, select the ADBC adapter bundle that is appropriate to the kind of data source your application will use. Click **Finish** or **OK**.

An **Adapters** node is added to your project if not already present. The node will contain a node for the new adapter. The adapter node contains an instance of the new adapter.

When you add a data source-specific adapter, the **ADBC Adapter common (Apama Database Connector Common common)** bundle will be added to the project automatically.


3. Configure the instance of the new adapter as described in "Configuring an ADBC adapter" in *Connecting Apama Applications to External Components*.

## Launching the project

To create queries most efficiently, the project needs to be running so that you can see what data sources, databases, and existing queries are available.

---

### To create a new run configuration

1. Select the project for which to create the run configuration.
2. In the Software AG Designer menu bar, select **Run > Run Configuration**.
3. In the Run Configuration wizard, select **Apama Application** and click the **New launch configuration** button (.
4. On the **Apama Project** tab, specify the following:
  - Specify the **Name** of the run configuration.
  - Select or accept the **Project**.
  - Select the **Enable DataPlayer** check box. If this box is not checked, the Query Editor and Data Player control are disabled.
  - Select **Generate time events from data** if you want the correlator to use external time events (starting the correlator with the `-xclock` option). The **Generate time events from data** check box is available only if you checked the **Enable DataPlayer** check

box. For details about the format of correlator `&TIME` events, see *Generating events that keep time* in *Developing Apama Applications*.

When **Generate time events from data** is checked, the time field specified in the Data Player playback query must be a float value that represents a number of seconds since the epoch. The data player transforms these values into **&TIME** events.

When **Generate time events from data** is disabled (unchecked), the Data Player's **Speed** and **Play to** controls are disabled.

5. Click **Run** to save and launch the project. Or click **Apply** to save the configuration without running the project.

## Specifying Data Player playback queries

You create and modify Data Player queries with the Data Player Query Editor. The information for the Data Player queries is stored in the project's `dataplayer_queries.xml` file.


---

### To create or modify a Data Player query

1. In the Project Explorer view, expand the project and then expand the `config` folder. Double click on the `dataplayer_queries.xml` file to open the Data Player Query Editor. If the project is running, you will be able to make selections from the **Datasource**, **Database**, and **Query** drop-down lists. If the project is not running, most of the controls are disabled.

If the project is not running the Data Player Query Editor will report that it is offline; in this mode, you cannot select Data Sources or Databases and all the controls are disabled.

If you are creating a new Data Player query, you can also right-click on the name of the project and select **New > Data Player Query** from the pop-up menu. Use this method if you are working on an imported project that does not yet contain a `dataplayer_queries.xml` file.

2. Click the **Add Query** button (  ). The New Query dialog appears.
3. Provide a name for the new Data Player query and click **OK**.
4. In the **General Settings** section, specify the following properties for the Data Player query:
  - **Description** — Provide a description of the Data Player query.
  - **Data Source** — Select an available Data Source from the drop down list.
  - **Database** — Select an available Database from the drop down list.
  - **Return Type** — The choices are `Native` or `Wrapped`. When `Native` is selected, each matching event will be sent as-is to the correlator. When `Wrapped` is selected, each matching event will be “wrapped” in a container event. The container event will be named using the event name. Event wrapping allows events to be sent to the correlator without triggering application listeners. A separate user monitor



can listen for wrapped events, modify the contained event, and reroute it such that application listeners can match on it.

For example, you want to map the event `Tick` from the namespace `com.apama.demo` via the adapter, and your definition of `Tick` is as follows:

```
event Tick{
    string symbol;
    integer qty;
    float price;
}
```

In order to define the event return type as **Wrapped - Return events in container event**, you have to define an event in the default package such as the following:

```
event HistoricalTick{
    integer queryId;
    string serviceId;
    float timestamp;
    com.apama.demo.Tick tick;
}
```

The `Tick` event will then be wrapped in a `HistoricalTick` event.

5. In the **Query** section, the **Use Query Template** check box specifies whether you want to use a Data Player Query Template (checked) or a Raw Query (unchecked).

If you are using a **Query Template**:

- a. Select a Data Player Query Template from the drop-down list. The choices are the canned Data Player queries supplied with the Apama installation: **findAll**, **findEarliest**, **findLatest**, and **getCount**. You can add your own Data Player Query Templates; see ["Creating Data Player query templates" on page 148](#).
  - b. If the Data Player Query Template uses replaceable parameters, they will be displayed in the **Name** and **Value** table in the **Query** section of the editor.
6. Double click a cell in the **Value** column and type the entry to use for the Data Player query, for example the name of a database table or column.

If you are using a **Raw Query**, in the **Query String** text area specify the statement to execute as follows:

- For ODBC and JDBC data sources, use SQL syntax.
- For an Apama Sim data source, use the following keywords (keywords are case insensitive):

`TIME` with `=`, `<`, `>`, `>=`, and `<=` comparators and the time. Use one or two `TIME` statements. If two `TIME` statements are used, only data that matches both statements will be returned.

- `INCLUDE` plus the event type to retrieve.
- `EXCLUDE` plus the event type to not retrieve.

Examples of this syntax are:

```
TIME=12345
TIME=12345; INCLUDE=(com.apama.something)
TIME=12345; INCLUDE=(com.apama.something); EXCLUDE=(com.apama.something.abc)
```

```
TIME>12345;TIME<20000
```

7. In the **Event Type** field, specify **unmapped-sql** when the database table contains an `eventString` column that contains Apama events with string value. However, if the database table contains individual columns that are mapped to event fields in the ADBC adapter, then specify the name of the specific event in the **Event Type** field.

The value that you specify in the **Event Type** field becomes the value for `_ADBCType` in the ADBC adapter.






Given the above example of the `Tick` event that has been mapped in the adapter, you would have to specify `com.apama.demo.Tick` as the event type.

8. In the **Time Column** field, specify the time column in your table. For example, if your table contains a `TransactionTime` column, then you could specify **TransactionTime** in the **Time Column** field. Ensure that the value in the time column is a float value that represents a number of seconds since the epoch.
9. In the **Advanced Settings** section, specify any **Extra Parameters**.
  - a. Right-click in the **Name** column and select **Add** from the pop up menu (or double click in the **Name** column).
  - b. Specify the **Name** of the extra parameter.
  - c. In the **Value** column, specify the value of the extra parameter.

10. Save the Data Player query.

When you save the Data Player query, the Data Player query name is added to the Data Player Control view's drop down query list.

The **Query Editor** tool bar contains the following buttons:

-  **Configure Credentials** — This displays the Credential for selected data source dialog where you add a user name and password for the current Data Player query if they are required for accessing the data source and select the scope of the credentials from a drop-down list. You can specify if the credentials apply globally, to a specific data source, or to a specific Data Player query. Credentials entered here are shared by the Data Player Query Editor and the Data Player.
-  **Clone Query** — Select this if you want to make a copy of the current Data Player query. The Clone Query dialog prompts you for the name of the copy. You can then edit the specifications for the Data Player query.
-  **Add Query** — Select this if you want to add a new Data Player query. The New Query dialog prompts you for the name of the new Data Player query.
-  **Rename Query** — Select this if you want to rename the Data Player query. The Rename Query dialog prompts for the new name of the Data Player query.
-  **Delete Query** — Select this if you want to delete the Data Player query. The Delete Query dialog prompts you to confirm that this is what you want to do.

## Data Player Control view

---

The Data Player Control view is enabled when the project is launched if **Enable DataPlayer** has been checked in the project's Run Configuration. The tab for this view is located in the lower right of the Apama Developer perspective next to the **Console** and **Tasks** views. If the Data Player Control view is not shown, select **Window > Show View > Data Player Control** from the Software AG Designer menu.

## Playback settings

The individual controls available on title bar of the Data Player Control view are:

- **Query** — The available queries are listed in the Query drop-down list on the title bar.
- **Preferences** — The Preferences dialog appears by clicking the **view Menu** down arrow at the right-hand side of the control's title bar. From the Preferences dialog you can select default Time Zone to use and the Date/Time Format to use during playback.
- **Speed** — Specify playback speed from the drop-down list; 10x is generally a good balance between speed and understandability.
- **Throttle** — This specifies the Data Player throttling period. The Data Player coalesces scenario update events sent to dashboards and other clients over a throttling period in order to enhance playback performance. The default setting is 5.0 seconds. You can change this setting to match your application. If data overwhelms the correlator or clients, increase the setting. If data arrives too slowly for clients to operate optimally, reduce the setting. Setting the value to 0.0 turns it off.

## Playback controls

The Data Player playback controls are:

- **Play** — Start the playback.
- **Stop** — Stop the playback.
- **Pause** — Pause the playback.
- **Step** — Plays back a single event at a time.
- **Play to** — Displays the Choose Date Time dialog so you can select a specified date and time at which the playback will pause.

## Playback status

The Data Player displays the following status information:

- **Total Events** — Shows the total events that have been played back during the current playback operation.

- **Last Event Time** — Shows the recorded time of the last event played back.
- **Last Event** — Shows the contents of the last event played back.

When a playback session has run to completion or if you click the **Stop** button and **Generate time events from data** has been enabled, the Data Player control displays the Query Done dialog. You can move the dialog box around if it covers up status information needed in order to decide whether to restart or stop the project.

- **Restart** — Click **Restart** if you want to run another playback session. You can then display the Data Player control again and start another playback session.
- **Stop** — Click **Stop** if you want to terminate all processes associated with the playback session.

## Creating Data Player query templates

If you want to create a new Data Player query template, you need to specify them in the project's query template file. The Data Player query template file is found in the project's `bundle_instance_files` folder and is named `ADBC-queryTemplates-SQL.xml` if you are using an ODBC or JDBC data source or `ADBC-queryTemplates-Sim.xml` if you are using a Sim data source.

### To add a Data Player query template

1. In the Project Explorer view, expand the project's `bundle_instance_files` folder and then expand either the `ADBC_for_ODBC`, `ADBC_for_JDBC`, or `ADBC_for_Sim` folder as appropriate.
2. Double click the query templates file, either `ADBC-queryTemplates-SQL.xml` or `ADBC-queryTemplates-Sim.xml`. The file opens in the text editor. The following is the standard `findLatest` Data Player query template for ODBC and JDBC data sources.

```
<query
  name="findLatest"
  description="Get the row with the latest time."
  implementationFunction="substitution"
  inputString="select * from ${TABLE_NAME} order by
    ${TIME_COLUMN_NAME} desc limit 1">
  <parameter
    description="Name of a table to query"
    name="TABLE_NAME"
    type="String"
    default=""/>
  <parameter
    description="Name of the time column"
    name="TIME_COLUMN_NAME"
    type="String"
    default="timeField"/>
</query>
```

3. The Data Player query templates are defined in the `<query>` element. The available attributes you can specify are:

- `name` — A short name that will be displayed in the Query Template drop-down list.
- `description` — A short description of the Data Player query.
- `implementationFunction` — Currently Data Player query templates support the substitution function. This allows a token to be used as a place holder in a SQL query.
- `inputString` — For ODBC and JDBC data sources, use standard SQL syntax, from the example above:

```
inputString="select * from ${TABLE_NAME} order by
            ${TIME_COLUMN_NAME} desc limit 1">
```

In the above example, `$(TABLE_NAME)` and `$(TIME_COLUMN_NAME)` are parameters that will be replaced with a database table and column name when you create a Data Player query from the template (see `parameter`, below).

For Sim data sources use the `TIME` keyword plus the comparators `=`, `>`, `<`, `>=`, `<=` plus the time. You can use one or two `TIME` expressions. Note, because you are adding the expression to an xml file, you need to use `>` and `<` instead of the characters `>` and `<`. Do not use spaces before or after `>` and `<`.

The `inputString` expression can also use the `INCLUDE`, and `EXCLUDE` keywords, for example:

```
TIME=12345;INCLUDE=(com.apama.something);EXCLUDE=(com.apama.something.abc)
TIME>12345;TIME<20000
```

- `parameter` — For ODBC and JDBC data sources you can specify parameters and default values for the Data Player query template. This allows you to create different Data Player queries from the same template that, for example, query different tables or different columns in the database. From the above example:

```
<parameter
  description="Name of a table to query"
  name="TABLE_NAME"
  type="String"
  default=""/>
<parameter
  description="Name of the time column"
  name="TIME_COLUMN_NAME"
  type="String"
  default="timeField"/>
```

When you create a Data Player query from a template that has the parameters shown above in the Data Player Query Editor, you will specify the names of the database table and column to use with the Data Player query.

4. Save the Data Player query template file when you finish modifying it.

## Command-line Data Player interface

When you are ready to test your application, you can use the Data Player command-line interface to write scripts and unit tests to exercise the API layers. In some cases, it

might be easier to play back events to the correlator using the command-line interface as compared to using the Data Player GUI in Software AG Designer.

To use the command-line interface to the Data Player, you must have already used the GUI interface in Software AG Designer to define Data Player queries and Data Player query configurations in Software AG Designer. When you use the command-line interface, you specify the Data Player query names and Data Player query configurations that you created in Software AG Designer.

For more information on using the Data Player command line interface, see "Using the Data Player command line interface" in *Deploying and Managing Apama Applications*.

# 9

## Setting the Apama Preferences

---

■ Showing the Apama-specific preferences .....	152
■ Apama .....	152
■ Adapters .....	153
■ Web Service .....	153
■ Catalogs .....	154
■ MonitorScript .....	154
■ Editor Colors .....	155
■ Editor Formatting .....	155
■ Editor Templates .....	156
■ MonitorScript Path Variables .....	157
■ Profiling and Logging .....	158
■ Query .....	159
■ Syntax Coloring .....	159
■ Scenarios .....	160
■ Workbench .....	161

This section provides information on the Apama-specific preferences.

## Showing the Apama-specific preferences

---

The Apama-specific preferences are set in the **Preferences** dialog box of Eclipse.

---

### To show the Apama-specific preferences

1. From the **Window** menu, choose **Preferences**.
2. In the tree of the resulting dialog box, expand the **Software AG** node and then the **Apama** node.

Different pages are provided for setting different types of preferences.

3. Select one of the pages in the tree and set the required options as described in the topics below.
4. Click **OK** to save your changes and to close the dialog box.

## Apama

---

The top-level page of the Apama preferences provides the following options:

### Dashboard Process Arguments

This group box allows you to specify arguments for the dashboard process. Note that these are arguments to the JVM and not to the Dashboard Viewer itself. The most common use for this is to pass Java system properties or specify the memory constraints. For example, the `-Dsun.java2d.noddraw=true` argument, which is provided by default, defines a specific font so that international characters are displayed properly.

When you click the **Variables** button, the Select Variable dialog is shown. This is a standard Eclipse dialog which allows you to select variables to be used as launch configuration parameters (see the Eclipse documentation for more information on this dialog). The following variables pertain to Apama:

Variable Name	Description
<code>apama_home</code>	The root directory of the Apama installation.
<code>apama_library_version</code>	The Apama library version.
<code>apama_work</code>	The directory for Apama user files.



### Hide "Terminate Launch" dialog box

By default, the Terminate Launch dialog box is shown when you stop a running Apama project and associated processes are still running. This dialog box allows you to close these processes manually. It also provides the **Do this automatically from now on** option. When you enable this option, all running processes are automatically closed each time you stop a running Apama project, and the Terminate Launch dialog box is no longer shown. In addition, the **Hide "Terminate Launch" dialog box** option is automatically enabled in the preferences. If the Terminate Launch dialog box is to be shown again, you have to disable this option in the preferences.

**Note:** You can always terminate a running Apama project using the corresponding button in the Console view, regardless of the setting of the **Hide "Terminate Launch" dialog box** option.

### Location of Apama log file for Software AG Designer

This text box informs you in which directory you can find the Apama log file for Software AG Designer. This is just for your information; the text box cannot be edited. If you want to go to this directory directly, click the **Open location for Apama log file** link which is shown below the text box.

## Adapters

---

The **Adapters** page of the Apama preferences pertains to Web Services Client adapters. It provides the following option:

### View tooltip

This option applies to the entries in the **WebServices Client Adapter** node in the Project Explorer. When this option is selected, more detailed information about an entry is shown in a pop-up when you move the mouse over the entry.

See "Using the Apama Web Services Client Adapter" in *Connecting Apama Applications to External Components* for more information on this type of adapter.

## Web Service

---

The **Web Service** page is available when you expand the **Adapters** node in the Apama preferences. You can use it to set the preferences for the Web Service Client adapters. The following options are available:

### Store locally

This defines the default setting for the WSDL savings options in the Web Service Client wizard in which you define the operations for the WSDL. See "Specify the Web Service and operation to use" in *Connecting Apama Applications to External Components*.

If **Store locally** is selected, the WSDL is stored in the project (in the .services folder). This may increase the performance. You can select one of the following:

- **Download all imports to local file system.** If selected, each XSD is stored as a file in the project.
- **Leave all imports as it is.** If selected, the XSD locations as specified in the WSDL file are used directly.

#### Open editor after wizard

If selected, the Web Service adapter editor is immediately invoked after you have added the adapter to the project. See also "Adding a Web Services Client adapter to an Apama project" in *Connecting Apama Applications to External Components*.

## Catalogs

---

The **Catalogs** page of the Apama preferences allows you to specify what catalogs are available to your workspace. You can add catalogs for bundles, blocks and functions that are not included as part of the standard Apama installation. You can also remove catalogs. For more information on catalogs, see "Using the Catalogs tab" in *Using Event Modeler* which is part of the *Developing Apama Applications* guide.

The tabs that are available on this page (**Bundle catalogs**, **Block catalogs**, or **Function catalogs**) all look alike. Each of these tabs provides the following buttons:

Command Button	Description
<b>Add</b>	<p>Add a catalog for bundles, blocks, or functions that are not included as part of the standard Apama installation. In the resulting dialog, you have to select the folder that you want to add.</p> <p>When you add block and function catalogs, they will be available when you work on a scenario in the Event Modeler.</p>
<b>Add Variable</b>	<p>Add variables to your build path. In the resulting dialog, you can select one or more variables which point to a folder. Or you can click the <b>Extend</b> button for a selected variable and then select an archive within the folder.</p>
<b>Remove</b>	<p>Remove the selected catalog.</p>

## MonitorScript

---

The **MonitorScript** page of the Apama preferences allows you to toggle the editor's assistance options. The following options are available:

**Auto Close Quotes**

If selected (default), a closing quotation mark (") is automatically added when you insert a quotation mark.

**Auto Close Brackets**

This pertains to these types of brackets:

[ ]

( )

If selected (default), a matching closing bracket is automatically added when you insert an opening bracket.

**Auto Close Braces**

This pertains to this type of braces:

{ }

If selected (default), a matching closing brace is automatically added when you insert an opening brace.

**Context Assistance**

If selected (default), context assistance is turned on. When you type an event name followed by a period (.), a list of the event parameters defined by that event type pops up. You can then choose the parameter that you want to insert in your application.

---

## Editor Colors

The **Editor Colors** page is available when you expand the **MonitorScript** node in the Apama preferences.

You can use this page to change the colors for various elements in the EPL editor, for example, for comments or keywords. The current colors are shown in the preview area at the bottom of the **Editor Colors** page.

To change the color for an element, expand a node in the **Element** list box (for example, the **MonitorScript** node), and then click the element. The current color of that element is then shown on the **Color** button. Click the **Color** button, and then choose the new color in the resulting dialog box.

If you want to define a text style (such as bold or italic) for the selected element, select the corresponding check box(es).

Each change is shown in the preview area at the bottom of the page.

---

## Editor Formatting

The **Editor Formatting** page is available when you expand the **MonitorScript** node in the Apama preferences.

You can use this page to define whether tabs or spaces are to be used in the EPL editor when you press the TAB key. The following options are available:

#### Insert tabs instead of spaces

If this check box is selected, a tab character is inserted when you press TAB. If this check box is not selected, spaces are inserted when you press TAB.

#### Number of spaces that represents an indentation

Applies when spaces are used instead of tabs. Defines the number of spaces that are inserted when you press TAB. You can change this value as needed.

## Editor Templates

The **Editor Templates** page is available when you expand the **MonitorScript** node in the Apama preferences.

EPL templates provide a way for you to define a short name for a longer pattern that you often specify in an EPL file. For example, Apama provides the `dict` template. When you enter `dict` followed by a space in the EPL editor, `dictionary<>` is automatically inserted.

Apama provides the following standard templates:

Name	Description	Pattern
<code>dict</code>	<code>dictionary&lt;&gt;</code>	<code>dictionary&lt;%\c&gt;</code>
<code>ona</code>	<code>on all</code>	<code>on all %\c</code>
<code>onall</code>	<code>on all</code>	<code>on all %\c</code>
<code>seq</code>	<code>sequence&lt;&gt;</code>	<code>sequence&lt;%\c&gt;</code>

`%\c` in the pattern defines the position of the cursor after the template content has been inserted in the editor. The **Template** area at the bottom of the **Editor Templates** page always shows the pattern of the template that is currently selected.

You can define additional templates as needed. All Apama projects can use all EPL templates. You can also define templates in one Software AG Designer installation and then export them for use in another Software AG Designer installation.

The following buttons are provided:

Button	Description
<b>New</b>	Add a new template. In the resulting dialog box, specify the following: the name of the template (that is, the name that you

Button	Description
	type in the editor), the description (that is, the template content that is to be inserted in the editor), and the pattern (that is, the cursor position). To define the pattern, you can just copy the information from the description and then insert <code>%\c</code> at the position where you want the cursor to appear.
<b>Edit</b>	Edit the selected template. In the resulting dialog box, you can change the description and pattern.
<b>Remove</b>	Remove the selected pattern.
<b>Import</b>	Import new templates that have been exported from another Software AG Designer installation. In the resulting dialog box, navigate to the file that contains the templates and select it.
<b>Export All</b>	Export all templates that are currently defined. In the resulting dialog box, specify the name of the file that is to contain these templates. The name <code>Abbreviations.txt</code> is offered by default. If possible, specify a path that is reachable from other Software AG Designer installations that need to import the templates.

---

## MonitorScript Path Variables

The **MonitorScript Path Variables** page of the Apama preferences allows you to define path variables that you can use in all of your Apama projects. These path variables are helpful when you share your project with other users. If the project is dependent on one or more external files, not all users might have the external files stored in the same location. You can then agree upon a path variable that defines the location of the external files, and each user can then define this path variable on this preferences page. See also ["Configuring the project build path" on page 54](#).

Apama provides the following standard path variables:

Variable Name	Description
<code>APAMA_HOME</code>	The directory in which Apama has been installed. This variable cannot be changed or deleted.
<code>APAMA_WORK</code>	The Apama work directory. Do not confuse this with your Eclipse workspace. This variable cannot be changed or deleted.

The following buttons are provided:

Button	Description
<b>New</b>	Add a new variable. In the resulting dialog box, specify the following: the name of the variable and the path. To define the path, you can click either <b>File</b> or <b>Folder</b> . Choose whatever is convenient for you and then navigate to the appropriate folder or EPL file and select it. If the variable represents a directory, you can use the same variable in the path for more than one file. If the variable represents a file, you can use that variable for only that file's path.
<b>Edit</b>	Edit the selected variable. In the resulting dialog box, you can change the name and path as described for the <b>New</b> button.
<b>Remove</b>	Remove the selected variable.

When you click the **OK** button of the Preferences dialog after you have added, edited or deleted a variable, a message dialog appears, informing you that the build path variables have changed and that a full build is recommended for the changes to take effect. Click **Yes** to ensure that the current project state is consistent with the new variable. If you are sure that the new variable does not affect the current project state, you can click **No** instead.

The variables that you define in the preferences are available for selection in the project properties. For further information, see ["Specifying dependencies for a multi-user project" on page 56](#).

---

## Profiling and Logging

---

The **Profiling and Logging** page of the Apama preferences pertains to Apama's EPL Profiler (see also ["Profiling EPL Applications" on page 129](#)). The following options are available:

### Switch to associated perspective when profiling or importing logs

When you launch a profiling session, a dialog appears by default, asking if you want to switch to the Apama Profiler perspective. You can change this behavior by selecting one of the following options buttons:

- **Always.** When selected, the Apama Profiler perspective is shown immediately. A dialog will not appear.
- **Never.** When selected, the Apama Profiler perspective is not used. A dialog will not appear.
- **Prompt** (default). When selected, a dialog appears, asking if you want to switch to the Apama Profiler perspective.

### Refreshing views

By default, the Apama profiler polls for data automatically (**Automatic Refresh**), using the refresh interval (in seconds) that is defined in the **Refresh interval** field and then updates the display in the Execution Statistics view. You can change the polling frequency by specifying the desired value in this field.

You can also change the way the profiler updates the data from a polling mode (**Automatic Refresh**) to an on-demand mode (**Manual Refresh**). The manual refresh is useful, for example, if you want to let your application run up to a certain point and then retrieve profile data, or if you want to inject a set of test events and then examine the profile deltas. To do the manual refresh, you click the corresponding button in the local toolbar of the Profiling Monitor view. See also ["Profiling Monitor view" on page 133](#).

## Query

---

The **Query** page of the Apama preferences does not contain any options. When you expand the corresponding node in the tree, a subnode is shown. See the following topic for more information.

## Syntax Coloring

---

The **Syntax Coloring** page is available when you expand the **Query** node in the Apama preferences.

You can use this page to manage the syntax coloring that is used when you edit the EPL source code for a query definition in the Query Designer. See also ["Editing source code in Query Designer " on page 93](#).

To change the color for a syntax element, click that element in the **Token Styles** list box. The current color of that element and also its background color is then shown on the **Color** and **Background** buttons. Click one of these buttons, and then choose the new color in the resulting dialog box.

If you want to define a text style (such as bold or italic) for the selected element, select the corresponding check box(es).

If you want to define a different font for the selected element, click the **Change** button that is shown next to the font name that is currently used. You can then define a different font in the resulting dialog box. You can also define a different font style or font size in that dialog. If you define a different color or effect in this dialog, however, this will be ignored.

## Scenarios

---

The **Scenarios** page of the Apama preferences pertains to the Event Modeler (see also "Developing Apama Applications in Event Modeler" in *Developing Apama Applications*). It provides the tabs described below.

### General

The **General** tab provides the following options:

#### State Graph / Wiring Graph

The options in the **State Graph** group box pertain to the Event Modeler's **Event Flow** tab. See also "Working in the Event Flow panel" in *Developing Apama Applications*.

The options in the **Wiring Graph** group box pertain to the Event Modeler's **Block Wiring** tab. See also "Using the Block Wiring tab" in *Developing Apama Applications*.

In both group boxes, you can set the following options:

- **Show grid.** If selected, a grid is shown in the tab.
- **Snap to grid.** If selected, snap-to-grid is enabled in the tab. This helps you align the items in the tab evenly.
- **Grid color.** The color that is currently used for the grid is shown on this box. Click this box if you want to define a different grid color. You can then choose a new color in the resulting dialog.
- **Grid spacing.** Specifies the spacing between the grid lines. If you want to increase or decrease the grid spacing, enter a new value in pixels.

### Undo / Redo

The **History size** defines the maximum number of edits that are to be kept in the undo history of the Event Modeler. You can use the standard Eclipse **Undo** command (or CTRL +Z) multiple times to undo several actions at once, up to the limit set with this option.

### Colors and Fonts

**Colors and Fonts** tab allows you to change the colors and fonts of most of the graphical elements of the Event Modeler display. It provides the following options:

#### Colors

The colors that are currently used for the different elements are shown in the **color** column. To change the color for an element, click on the color that is shown for that element. In the resulting dialog, you can then pick a different color.

#### Fonts

The fonts that are currently used for the different elements are shown in the **font** column. To change the font for an element, click on the font description that is shown for that element. In the resulting dialog, you can then pick a different font.



## Workbench

---

The **Workbench** page of the Apama preferences allows you to customize the Workbench Project View (see also "[Workbench Project view](#)" on page 16). It provides the following options:

### Project view background color

The background color that is currently used in the Workbench Project View is shown on the button. Click this button if you want to define a different background color. You can then choose a new color in the resulting dialog.

### Project view filters

When you define a filter, you specify the pattern that determines which files or folders are to be hidden in the Workbench Project View. By default, all files starting with a period ( `.` ) and Java class files ( `*.class` ) are filtered out.

The following buttons are provided:

Button	Description
<b>Add</b>	Add a new filter. Specify the pattern for the filter in the resulting dialog. You can use the wildcards <code>*</code> (any number of characters) and <code>?</code> (any single character) in the pattern.
<b>Remove</b>	Remove the selected filter.

If you want to change the pattern for a filter, you have to remove the filter and then add it once more.

When you click  in the Workbench Project View to show all folders, the folders and files for which you have defined filter patterns are not shown.